

# **A comparison of the impact of data vault and dimensional modelling on data warehouse performance and maintenance**

**M van Schalkwyk  
12302619**

Dissertation submitted in partial fulfilment of the requirements  
for the degree *Magister Scientiae* in Computer Science at the  
Potchefstroom Campus of the North-West University

Supervisor: Dr R Goede

May 2014



# Abstract

**Keywords:** Data warehouse, dimensional modelling, data vault modelling, data warehouse modelling techniques, Kimball, Linstedt, query performance, load performance, ETL performance, flexibility, storage requirements.

This study compares the impact of dimensional modelling and data vault modelling on the performance and maintenance effort of data warehouses. Dimensional modelling is a data warehouse modelling technique pioneered by Ralph Kimball in the 1980s that is much more effective at querying large volumes of data in relational databases than third normal form data models. Data vault modelling is a relatively new modelling technique for data warehouses that, according to its creator Dan Linstedt, was created in order to address the weaknesses of dimensional modelling. To date, no scientific comparison between the two modelling techniques have been conducted.

A scientific comparison was achieved in this study, through the implementation of several experiments. The experiments compared the data warehouse implementations based on dimensional modelling techniques with data warehouse implementations based on data vault modelling techniques in terms of load performance, query performance, storage requirements, and flexibility to business requirements changes.

An analysis of the results of each of the experiments indicated that the data vault model outperformed the dimensional model in terms of load performance and flexibility. However, the dimensional model required less storage space than the data vault model. With regards to query performance, no statistically significant differences existed between the two modelling techniques.

# Uittreksel

**Sleutelwoorde:** Datapakhuis, dimensionele modellering, datakluismodellering, datapakhuismodellering, Kimball, Linstedt, navraagwerkverrigting, laaiwerkverrigting, aanpasbaarheid, stoorvereistes.

In hierdie studie word die impak van dimensionele modellering en datakluismodellering op die werkverrigting en die instandhouding van datapakhuise vergelyk. Dimensionele modellering is 'n datapakhuismodelleringstegniek wat deur Ralph Kimball in die 1980's ontwikkel is en baie meer effektief is met databasis navrae oor groot volumes data as derde normaalvorm datamodelle. Datakluismodellering is 'n relatief nuwe modelleringstegniek vir datapakhuise wat, volgens die ontwerper, Dan Linstedt, geskep is om die swakpunte van dimensionele modellering aan te spreek. Tot dusver, is daar nog geen wetenskaplike vergelyking tussen die twee modelleringstegnieke gedoen nie.

'n Wetenskaplike vergelyking word in hierdie studie gedoen deur die uitvoering van verskeie eksperimente. Die eksperimente vergelyk implementerings van datapakhuise wat op dimensionele modellerings tegnieke gebaseer is met implementerings van datapakhuise wat op datakluismodelleringstegnieke gebaseer is in terme van laaiwerkverrigting, navraagwerkverrigting, stoorvereistes en aanpasbaarheid om aan veranderende behoeftes te voldoen.

'n Ontleding van die resultate van die eksperimente het getoon dat die datakluismodel beter gevaar het as die dimensionele model in terme van laaiwerkverrigting en aanpasbaarheid. Maar die dimensionele model vereis minder stoorspasie as die datakluismodel. Met betrekking tot navraagwerkverrigting was daar geen statisties beduidende verskille tussen die twee modelleringstegnieke nie.

# Acknowledgement

The research and writing of this dissertation has been both challenging and rewarding. I take immense pride in the final result, but need to acknowledge the many people that made this possible. In particular, I would like to thank the following persons:

- My supervisor, Prof. Roelien Goede for her patience and guidance throughout this process.
- Van Zyl Kruger and Martijn Evers for reviewing my dimensional and data vault models respectively. Approval from my peers adds validity to my research.
- Dan E. Linstedt, the creator of data vault modelling, who was instrumental in the acquisition of my data vault modelling skills.
- A special thanks to both my parents, Pieter and Mariëtte van Schalkwyk, for their continued support and motivation.
- Most importantly, my wife Elmarie and my two children, Zané and Zian. Without their love, support and sacrifices over the past several years I would not have been able to complete this dissertation.
- My Father in heaven without whom none of this would have been possible.

*In loving memory of Gawie van Rensburg who would have been so proud.*

# Contents

|                                                        |      |
|--------------------------------------------------------|------|
| Abstract.....                                          | i    |
| Uittreksel .....                                       | ii   |
| Acknowledgement .....                                  | iii  |
| Contents.....                                          | iv   |
| List of tables .....                                   | ix   |
| List of figures.....                                   | xiii |
| Chapter 1: Introduction .....                          | 1    |
| 1.1    Introduction .....                              | 1    |
| 1.2    Problem statement and substantiation .....      | 1    |
| 1.3    Research aims .....                             | 3    |
| 1.4    Research objective and sub-objectives .....     | 3    |
| 1.4.1    Load performance .....                        | 3    |
| 1.4.2    Query performance .....                       | 3    |
| 1.4.3    Storage space required .....                  | 3    |
| 1.4.4    Flexibility .....                             | 3    |
| 1.5    Empirical investigation.....                    | 4    |
| 1.5.1    Design and method .....                       | 4    |
| 1.5.2    Measuring instruments .....                   | 4    |
| 1.5.3    Validity procedures and quality control ..... | 5    |
| 1.5.4    Proposed method of data processing .....      | 5    |
| 1.5.5    Expected outcomes.....                        | 5    |
| 1.6    Chapter division .....                          | 5    |
| 1.7    Summary.....                                    | 6    |
| Chapter 2: Research methodology .....                  | 7    |
| 2.1    Introduction .....                              | 7    |
| 2.2    Research terminology .....                      | 7    |
| 2.3    Research paradigms .....                        | 10   |
| 2.3.1    Positivism.....                               | 11   |

|                                          |                                                   |    |
|------------------------------------------|---------------------------------------------------|----|
| 2.3.2                                    | Interpretive research .....                       | 12 |
| 2.3.3                                    | Critical social research .....                    | 13 |
| 2.3.4                                    | Position of this study .....                      | 14 |
| 2.4                                      | Hypothetico-deductive method .....                | 14 |
| 2.5                                      | Experiments .....                                 | 15 |
| 2.5.1                                    | Experimental design.....                          | 16 |
| 2.5.2                                    | The five-step method for true experiments .....   | 18 |
| 2.5.3                                    | Hypothesis .....                                  | 19 |
| 2.5.4                                    | Validity .....                                    | 19 |
| 2.6                                      | Qualitative or quantitative research? .....       | 20 |
| 2.7                                      | Summary.....                                      | 22 |
| Chapter 3: Data warehouse modelling..... |                                                   | 23 |
| 3.1                                      | Introduction .....                                | 23 |
| 3.2                                      | Definitions .....                                 | 24 |
| 3.3                                      | Data warehouse modelling techniques .....         | 25 |
| 3.4                                      | Running example .....                             | 25 |
| 3.5                                      | Dimensional modelling .....                       | 33 |
| 3.5.1                                    | Facts and fact tables .....                       | 40 |
| 3.5.2                                    | Attributes and dimension tables .....             | 41 |
| 3.5.3                                    | Four-step process for dimensional modelling ..... | 43 |
| 3.5.4                                    | Conformed dimensions and stove pipes.....         | 46 |
| 3.5.5                                    | Snowflaking.....                                  | 47 |
| 3.5.6                                    | Slowly changing dimensions .....                  | 47 |
| 3.5.7                                    | Many-to-many dimension relationships .....        | 50 |
| 3.5.8                                    | Ragged hierarchies .....                          | 51 |
| 3.5.9                                    | Dimensional modelling indexing strategies.....    | 52 |
| 3.6                                      | Data vault modelling.....                         | 53 |
| 3.6.1                                    | Hub entities .....                                | 68 |
| 3.6.2                                    | Satellite entities .....                          | 71 |
| 3.6.3                                    | Link entities .....                               | 74 |

|            |                                                                                                |     |
|------------|------------------------------------------------------------------------------------------------|-----|
| 3.6.4      | Link-to-link relationships .....                                                               | 82  |
| 3.6.5      | Bridge tables .....                                                                            | 83  |
| 3.6.6      | Cross-reference tables .....                                                                   | 87  |
| 3.6.7      | Four-step (or seven-step) process to data vault modelling .....                                | 88  |
| 3.6.8      | Data vault indexing strategies .....                                                           | 89  |
| 3.7        | Comparison between dimensional and data vault modelling techniques .....                       | 90  |
| 3.7.1      | Smart keys .....                                                                               | 90  |
| 3.7.2      | Similarities between fact tables and transactional links .....                                 | 90  |
| 3.7.3      | Similarities between slowly changing dimensions and satellite entities.....                    | 91  |
| 3.7.4      | The number of joins and expected query performance.....                                        | 92  |
| 3.7.5      | Flexibility .....                                                                              | 93  |
| 3.8        | Correlation between data warehouse modelling technique and data warehouse project success..... | 95  |
| 3.9        | Summary.....                                                                                   | 96  |
| Chapter 4: | Experiment.....                                                                                | 98  |
| 4.1        | Introduction .....                                                                             | 98  |
| 4.2        | Overview of the experiments .....                                                              | 98  |
| 4.3        | Testing server .....                                                                           | 99  |
| 4.3.1      | Database management system (DBMS) .....                                                        | 99  |
| 4.3.2      | Operating system .....                                                                         | 100 |
| 4.3.3      | Eliminating uncontrolled variables .....                                                       | 100 |
| 4.4        | The dataset .....                                                                              | 100 |
| 4.4.1      | Dataset requirements .....                                                                     | 100 |
| 4.4.2      | Dataset candidates .....                                                                       | 101 |
| 4.4.3      | The TPC-E dataset .....                                                                        | 103 |
| 4.4.4      | TPC-E data model.....                                                                          | 104 |
| 4.5        | The user requirements .....                                                                    | 107 |
| 4.6        | Target data models .....                                                                       | 108 |
| 4.6.1      | Dimensional model.....                                                                         | 109 |
| 4.6.2      | Data vault model .....                                                                         | 114 |



|         |                                    |     |
|---------|------------------------------------|-----|
| 4.7     | Experiments .....                  | 118 |
| 4.7.1   | Experimental design.....           | 119 |
| 4.7.2   | Statistical analysis methods ..... | 120 |
| 4.7.3   | Load performance .....             | 120 |
| 4.7.3.1 | Define the research question.....  | 120 |
| 4.7.3.2 | Formulate the hypotheses.....      | 121 |
| 4.7.3.3 | Predict the results .....          | 121 |
| 4.7.3.4 | Execute the experiment.....        | 123 |
| 4.7.3.5 | Results and analysis .....         | 124 |
| 4.7.3.6 | Conclusion .....                   | 128 |
| 4.7.4   | Query performance .....            | 129 |
| 4.7.4.1 | Define the research question.....  | 129 |
| 4.7.4.2 | Formulate the hypothesis .....     | 129 |
| 4.7.4.3 | Predict the results .....          | 130 |
| 4.7.4.4 | Execute the experiment.....        | 131 |
| 4.7.4.5 | Results and analysis .....         | 131 |
| 4.7.4.6 | Conclusion .....                   | 136 |
| 4.7.5   | Storage size .....                 | 136 |
| 4.7.5.1 | Define the research question.....  | 137 |
| 4.7.5.2 | Formulate the hypothesis .....     | 137 |
| 4.7.5.3 | Predict the results .....          | 137 |
| 4.7.5.4 | Execute the experiment.....        | 139 |
| 4.7.5.5 | Results and analysis .....         | 139 |
| 4.7.5.6 | Conclusion .....                   | 145 |
| 4.7.6   | Flexibility .....                  | 145 |
| 4.7.6.1 | Define research question .....     | 145 |
| 4.7.6.2 | Formulate the hypothesis .....     | 145 |
| 4.7.6.3 | Predict the results .....          | 146 |
| 4.7.6.4 | Execute the experiment.....        | 146 |
| 4.7.6.5 | Results and analysis .....         | 153 |

|            |                                    |     |
|------------|------------------------------------|-----|
| 4.7.6.6    | Conclusion .....                   | 154 |
| 4.8        | Summary.....                       | 154 |
| Chapter 5: | Conclusions .....                  | 156 |
| 5.1        | Introduction .....                 | 156 |
| 5.2        | Summary of experiment results..... | 156 |
| 5.2.1      | Load performance .....             | 156 |
| 5.2.2      | Query performance .....            | 157 |
| 5.2.3      | Storage size .....                 | 158 |
| 5.2.4      | Flexibility .....                  | 159 |
| 5.3        | Limitations.....                   | 159 |
| 5.4        | Future research.....               | 160 |
| 5.5        | Chapter summary.....               | 161 |
| References | .....                              | 164 |
| Annexures  | .....                              | 170 |

## List of tables

|                                                                                                                                      |    |
|--------------------------------------------------------------------------------------------------------------------------------------|----|
| Table 2.1 Summary of research terminology by different writers as well as the terminology and definitions adopted in this study..... | 9  |
| Table 3.1 Mr DVD sample data (Category).....                                                                                         | 28 |
| Table 3.2 Mr DVD sample data (JobTitle).....                                                                                         | 28 |
| Table 3.3 Mr DVD sample data (RoleType).....                                                                                         | 28 |
| Table 3.4 Mr DVD sample data (Actor).....                                                                                            | 29 |
| Table 3.5 Mr DVD sample data (ProductionCompany) .....                                                                               | 29 |
| Table 3.6 Mr DVD sample data (Employee) .....                                                                                        | 30 |
| Table 3.7 Mr DVD sample data (DVDActor) .....                                                                                        | 31 |
| Table 3.8 Mr DVD sample data (Customer).....                                                                                         | 32 |
| Table 3.9 Mr DVD sample data (DVD).....                                                                                              | 32 |
| Table 3.10 Mr DVD sample data (Order).....                                                                                           | 33 |
| Table 3.11 Mr DVD sample data (OrderLineItem).....                                                                                   | 33 |
| Table 3.12 Mr DVD sample (Dim_Customer) .....                                                                                        | 35 |
| Table 3.13 Mr DVD sample data (Dim_DVD) .....                                                                                        | 35 |
| Table 3.14 Mr DVD sample data (Bridge_DVD_Actor) .....                                                                               | 36 |
| Table 3.15 Mr DVD sample data (Bridge_Org_Struc).....                                                                                | 37 |
| Table 3.16 Mr DVD sample data (Dim_Employee) .....                                                                                   | 37 |
| Table 3.17 Mr DVD sample data (Dim_Date) .....                                                                                       | 38 |
| Table 3.18 Mr DVD sample data (Dim_Actor).....                                                                                       | 38 |
| Table 3.19 Mr DVD sample data (Dim_ProdCom).....                                                                                     | 38 |
| Table 3.20 Mr DVD sample data (Fact_Sales) .....                                                                                     | 39 |
| Table 3.21 Comparison of facts and dimensions .....                                                                                  | 43 |
| Table 3.22 Enterprise data warehouse bus matrix example .....                                                                        | 45 |
| Table 3.23 Sample source data: sales amount per month, store and manager .....                                                       | 48 |
| Table 3.24 Type 2 slowly changing dimension before changes .....                                                                     | 49 |
| Table 3.25 Type 2 slowly changing dimension after changes .....                                                                      | 49 |
| Table 3.26 Fact_Sales using type 2 slowly changing dimension.....                                                                    | 49 |

|                                                                                           |    |
|-------------------------------------------------------------------------------------------|----|
| Table 3.27 Mr DVD sample data (HUB_Actor) .....                                           | 55 |
| Table 3.28 Mr DVD sample data (SAT_Actor) .....                                           | 55 |
| Table 3.29 Mr DVD sample data (HUB_Cat) .....                                             | 56 |
| Table 3.30 Mr DVD sample data (SAT_Cat).....                                              | 56 |
| Table 3.31 Mr DVD sample data (HUB_DVD) .....                                             | 56 |
| Table 3.32 Mr DVD sample data (SAT_DVD).....                                              | 57 |
| Table 3.33 Mr DVD sample data (HUB_Emp) .....                                             | 57 |
| Table 3.34 Mr DVD sample data (SAT_Emp) .....                                             | 57 |
| Table 3.35 Mr DVD sample data (HUB_JobTitle) .....                                        | 58 |
| Table 3.36 Mr DVD sample data (SAT_JobTitle) .....                                        | 58 |
| Table 3.37 Mr DVD sample data (HUB_ProdCom).....                                          | 58 |
| Table 3.38 Mr DVD sample data (SAT_ProdCom) .....                                         | 58 |
| Table 3.39 Mr DVD sample data (HUB_RoleType).....                                         | 58 |
| Table 3.40 Mr DVD sample data (SAT_RoleType) .....                                        | 59 |
| Table 3.41 Mr DVD sample data (HUB_Cust) .....                                            | 59 |
| Table 3.42 Mr DVD sample data (SAT_Cust_Addr_Ph) .....                                    | 60 |
| Table 3.43 Mr DVD sample data (SAT_Cust_Name_Gender).....                                 | 60 |
| Table 3.44 Mr DVD sample data (XREF_Date) .....                                           | 61 |
| Table 3.45 Mr DVD sample data (HLNK_Emp_Man).....                                         | 61 |
| Table 3.46 Mr DVD sample data (LNK_DVD_Actor_RoleType).....                               | 62 |
| Table 3.47 Mr DVD sample data (LNK_DVD_Cat) .....                                         | 62 |
| Table 3.48 Mr DVD sample data (LNK_DVD_ProdCom).....                                      | 63 |
| Table 3.49 Mr DVD sample data (LNK_Emp_JobTitle).....                                     | 63 |
| Table 3.50 Mr DVD sample data (LNK_Order) .....                                           | 64 |
| Table 3.51 Mr DVD sample data (LNK_OrderLine).....                                        | 65 |
| Table 3.52 Example of changing data in a satellite table .....                            | 72 |
| Table 3.53 Illustrating rate of change in satellite table .....                           | 73 |
| Table 3.54 Normalising attributes by rate of change .....                                 | 74 |
| Table 3.55 Normalising attributes by rates of change .....                                | 74 |
| Table 3.56 Example of HUB_ProdCom for handling collisions on the production company ..... | 79 |

|                                                                                           |     |
|-------------------------------------------------------------------------------------------|-----|
| Table 3.57 Example of same-as link for handling collisions on the production company..... | 79  |
| Table 3.58 DVDs by category and production company .....                                  | 84  |
| Table 3.59 Sample bridge table data .....                                                 | 86  |
| Table 3.60 Summary of how the changed business rules impacted the data models .....       | 95  |
| Table 4.1 TPC-E trade types (Anon., 2010:47).....                                         | 103 |
| Table 4.2 TPC-E table groupings (Anon., 2010:43) .....                                    | 104 |
| Table 4.3 Summary of subset of TPC-E tables.....                                          | 106 |
| Table 4.4 Historical tracking requirement.....                                            | 108 |
| Table 4.5 Dimensional model system fields summary .....                                   | 113 |
| Table 4.6 Data vault model resulting link entities .....                                  | 116 |
| Table 4.7 Data vault model resulting satellite entities .....                             | 117 |
| Table 4.8 Sample raw result set for history data load experiment.....                     | 124 |
| Table 4.9 Historical load results in seconds.....                                         | 125 |
| Table 4.10 Incremental load results in seconds.....                                       | 126 |
| Table 4.11 History load results – Wilcoxon matched pairs test .....                       | 127 |
| Table 4.12 History load results – T-test for dependent samples .....                      | 127 |
| Table 4.13 Incremental load results: t-test for dependent samples .....                   | 128 |
| Table 4.14 Query categories in terms of dimensional models.....                           | 129 |
| Table 4.15 Category 1 query comparison.....                                               | 130 |
| Table 4.16 Mean query duration in seconds by category and target data model.....          | 131 |
| Table 4.17 Category 1 query results, t-test for dependent samples .....                   | 132 |
| Table 4.18 Category 2 query results, t-test for dependent samples .....                   | 132 |
| Table 4.19 Category 3 query results, t-test for dependent samples .....                   | 132 |
| Table 4.20 Category 4 query results, t-test for dependent samples .....                   | 132 |
| Table 4.21 Category 5 query results, t-test for dependent samples .....                   | 133 |
| Table 4.22 All categories query results, t-test for dependent samples .....               | 133 |
| Table 4.23 Mandatory auditing fields .....                                                | 138 |
| Table 4.24 Storage size experiment results in MB.....                                     | 139 |
| Table 4.25 Data size results, t-test for dependent samples .....                          | 139 |
| Table 4.26 Index size results, t-test for dependent samples .....                         | 140 |

|                                                                                |     |
|--------------------------------------------------------------------------------|-----|
| Table 4.27 Total size results, t-test for dependent samples .....              | 140 |
| Table 4.28 Storage predictions after an additional 100 incremental loads ..... | 144 |
| Table 4.29 Flexibility experiment results .....                                | 154 |
| Table 5.1 Query categories in terms of dimensional models.....                 | 158 |

# List of figures

|                                                                                              |    |
|----------------------------------------------------------------------------------------------|----|
| Figure 2.1 The scientific method (Dodig-Crnkovic, 2002:3) .....                              | 15 |
| Figure 2.2 Experimental design notation .....                                                | 17 |
| Figure 2.3 Design type decision tree (Trochim, 2012) .....                                   | 18 |
| Figure 2.4 Research paradigm and classification of data (Straub et al., 2005) .....          | 21 |
| Figure 3.1 Example of employee hierarchy.....                                                | 26 |
| Figure 3.2 Mr DVD entity relationship diagram .....                                          | 27 |
| Figure 3.3 Dimensional data model for Mr DVD example .....                                   | 34 |
| Figure 3.4 Star schema or dimensional model.....                                             | 40 |
| Figure 3.5 Time dimension hierarchy.....                                                     | 42 |
| Figure 3.6 Conformed time and store dimensions .....                                         | 46 |
| Figure 3.7 Snowflaked date dimension.....                                                    | 47 |
| Figure 3.8 Type 2 slowly changing dimension model.....                                       | 48 |
| Figure 3.9 Mr DVD type 3 slowly changing dimension example.....                              | 50 |
| Figure 3.10 Bridge table example for resolving many-to-many dimension relationships .....    | 51 |
| Figure 3.11 Bridge table example implementing ragged hierarchies.....                        | 52 |
| Figure 3.12 Data vault data model for the Mr DVD example .....                               | 54 |
| Figure 3.13 Comparison of batch and real-time timestamps (Linstedt & Graziano, 2010:31) .... | 67 |
| Figure 3.14 Hub entity mandatory components and HUB_Emp example .....                        | 68 |
| Figure 3.15 Changing business keys across lines of business .....                            | 70 |
| Figure 3.16 Satellite entity structure and an example of a satellite .....                   | 71 |
| Figure 3.17 Satellite normalisation by source system (Linstedt & Graziano, 2010:117) .....   | 73 |
| Figure 3.18 Mandatory components of link entities and LNK_DVD_Actor_RoleType example         | 75 |
| Figure 3.19 Enforcing one-to-many or many-to-one relationships between hubs.....             | 76 |
| Figure 3.20 Implementing many-to-many relationships between hubs through links.....          | 76 |
| Figure 3.21 Example of mine section hierarchy .....                                          | 77 |
| Figure 3.22 Example hierarchical link data vault model .....                                 | 77 |
| Figure 3.23 Example of collisions on a production company business key .....                 | 78 |
| Figure 3.24 Example of a data vault model for handling collisions on the production company  | 79 |

|                                                                                                 |     |
|-------------------------------------------------------------------------------------------------|-----|
| Figure 3.25 Transactional link data vault model example .....                                   | 80  |
| Figure 3.26 Changes to satellite for transactional links .....                                  | 81  |
| Figure 3.27 Collapsing the satellite into the link for transactional links .....                | 81  |
| Figure 3.28 Link-to-link (parent-child) relationship example for logical data vault model ..... | 82  |
| Figure 3.29 Denormalised link-to-link (parent-child) relationships .....                        | 83  |
| Figure 3.30 Bridge table structure and example .....                                            | 83  |
| Figure 3.31 HUB_DVD, HUB_Cat and HUB_ProdCom with related link entities .....                   | 84  |
| Figure 3.32 SQL query to data vault without bridge table .....                                  | 85  |
| Figure 3.33 SQL query utilising bridge table to reduce the number of joins .....                | 87  |
| Figure 3.34 Example fact table (left) and type 2 transactional link (right) .....               | 91  |
| Figure 3.35 Comparing satellite and type 2 slowly changing dimension .....                      | 92  |
| Figure 3.36 DVD dimension containing DVD related information .....                              | 92  |
| Figure 3.37 Data vault model containing DVD related information .....                           | 93  |
| Figure 3.38 Data vault model before and after change .....                                      | 94  |
| Figure 3.39 Dim_DVD before (left) and after (right) change .....                                | 95  |
| Figure 4.1 TPC-E entity relationship diagram .....                                              | 105 |
| Figure 4.2 Dimensional model ERD - Trade fact table .....                                       | 110 |
| Figure 4.3 Dimensional model ERD: company performance fact table .....                          | 111 |
| Figure 4.4 Dimensional model ERD: market performance fact table .....                           | 112 |
| Figure 4.5 Data vault model ERD .....                                                           | 115 |
| Figure 4.6 History load experimental design .....                                               | 119 |
| Figure 4.7 Incremental load experimental design .....                                           | 120 |
| Figure 4.8 History load results: box-and-whisker plot .....                                     | 127 |
| Figure 4.9 Incremental load results: box-and-whisker plot .....                                 | 128 |
| Figure 4.10 Category 1 query results, box-and-whisker plot .....                                | 133 |
| Figure 4.11 Category 2 query results, box and whisker plot .....                                | 134 |
| Figure 4.12 Category 3 query results, box-and-whisker plot .....                                | 134 |
| Figure 4.13 Category 4 query results, box-and-whisker plot .....                                | 135 |
| Figure 4.14 Category 5 query results, box-and-whisker plot .....                                | 135 |
| Figure 4.15 All categories query results, box-and-whisker plot .....                            | 136 |



|                                                                                            |     |
|--------------------------------------------------------------------------------------------|-----|
| Figure 4.16 Data size results, box-and-whisker plot .....                                  | 140 |
| Figure 4.17 Index size results, box-and-whisker plot .....                                 | 141 |
| Figure 4.18 Total size results, box-and-whisker plot .....                                 | 141 |
| Figure 4.19 Size results, scatterplot of data size over time .....                         | 142 |
| Figure 4.20 Size results, scatterplot of index size over time .....                        | 143 |
| Figure 4.21 Size results, scatterplot of total size over time .....                        | 143 |
| Figure 4.22 Storage predictions after an additional 100 incremental loads .....            | 144 |
| Figure 4.23 Changed customer account relationship .....                                    | 147 |
| Figure 4.24 Original customer account relationship .....                                   | 147 |
| Figure 4.25 Changed industry company relationship .....                                    | 147 |
| Figure 4.26 Original industry company relationship .....                                   | 148 |
| Figure 4.27 New broker type requirement.....                                               | 148 |
| Figure 4.28 Customer account relationship change in dimensional model.....                 | 149 |
| Figure 4.29 Many-to-many relationship between company and industry with bridge table ..... | 150 |
| Figure 4.30 Many-to-many dimensional relationship between company and industry .....       | 151 |
| Figure 4.31 Many-to-many data vault relationship between company and industry .....        | 152 |
| Figure 4.32 Broker-type change .....                                                       | 152 |
| Figure 4.33 Original D_Broker dimension .....                                              | 153 |
| Figure 4.34 Broker type change to data vault model.....                                    | 153 |

# **Chapter 1: Introduction**

## **1.1 Introduction**

A data warehouse is defined as a subject-oriented, integrated, non-volatile, time-variant collection of data with the ultimate goal of decision support (Inmon, Strauss, and Neushloss, 2008:7). Data warehouse modelling refers to the data modelling technique which is used to represent/describe the data in the data warehouse. It also determines the physical implementation of the warehouse.

The aim of this research study was to apply a relatively new data warehouse modelling technique, termed “data vault modelling”, and to compare this technique with dimensional modelling in terms of performance and maintenance. Dan Linstedt, the creator of data vault modelling, claims that this new technique has certain advantages when compared to dimensional modelling, especially with regards the speeds at which data can be loaded into this model and also its flexibility (Linstedt, Graziano & Hultgren, 2009a:56).

The aim of the chapter is to provide evidence of the need for research that compares the impact of data vault modelling as opposed to dimensional modelling on data warehouse performance and maintenance. The chapter outlines the problem statement, the research aims and objectives and the method of investigation employed in the study.

Section 1.2 presents the problem statement and substantiation. Section 1.3 describes the aim of the research while section 1.4 discusses several hypotheses that will be tested in order to meet the research objectives. Section 1.5 introduces the research design and the method of empirical investigation, the instruments that will be used to measure the results, the measures that will be taken to ensure the validity of the results as well as the expected outcomes of the study. The chapter concludes with an outline of the remaining chapters and a summary of this chapter.

## **1.2 Problem statement and substantiation**

Data modelling for data warehousing is a fairly recent concept in the world of information technology and dates back to the early 1980s (Linstedt, 2002). As data warehousing grows in maturity, new demands are continuously being placed on data warehouses with these

increased demands exposing flaws in the existing methods of data warehouse modelling. Third normal form (3NF) data models have been used for online transactional processing (OLTP) systems since the early 1960s (Linstedt *et al.*, 2009a:69). One of the characteristics of OLTP systems is extremely quick data loads (Inmon, 2002:27). These extremely quick data loads arise from the fact that 3NF models, through normalisation, have no data redundancy (Kimball, Ross, Thornthwaite, Mundy & Becker, 2008:236). One of the biggest concerns regarding OLTP systems was their inability to store changes to data over time (Inmon, 2002:15). Nevertheless, Inmon solved this problem by creating an enterprise data warehouse (EDW), also known as the Corporate Information Factory (CIF). The data model for the EDW was, essentially, a 3NF model with a date-time stamp added to the primary key of each table, which allowed for the storage of data over time (Linstedt, 2002). However, the EDW model lacked query performance (Kimball, Reeves, Ross & Thornthwaite, 1998:142) and flexibility to change (Linstedt *et al.*, 2009a:11). Accordingly, in order to address this problem, Ralph Kimball designed dimensional modelling. Dimensional modelling brought with it several benefits, including improved query performance through denormalisation (Inmon, 2002:105) and greater flexibility to changes in the model, which Kimball termed graceful changes (Kimball, 2001). However, with the recent dramatic increase in data volumes (Anon, 2005), dimensional modelling has also started to show some deficiencies in terms of loading performance (Linstedt *et al.*, 2009a:11). With the increased amounts of data, it is, thus, essential that loading performance improve in order to load data within the available batch window.

Data vault modelling is a data warehouse modelling technique which was developed by Dan Linstedt and which attempts to address, inter alia, the issue of load performance (Linstedt *et al.*, 2009a:56). Linstedt also claims that the model is more resilient and flexible to change than dimensional modelling (Linstedt & Graziano, 2010:67). However, Karl Popper would regard these claims as myth because they have not yet been scientifically proven (Chua, 1986:607).

This research study aims to validate claims from industry and, thus, add to the existing body of knowledge in respect of the data warehouse modelling field. In view of the fact that data vault modelling is a recent and, as yet, relatively unknown development, favourable research results may lead to a positive attitude to the adoption of this modelling technique. This should, in turn, lead to increased data vault implementations and this would resolve some of the problems that dimensional modelling implementations are currently facing.

## **1.3 Research aims**

The aim of this research study was to scientifically compare data warehouse implementations based on dimensional modelling techniques with data warehouse implementations based on data vault modelling techniques in terms of performance and maintenance by using objective measurements.

## **1.4 Research objective and sub-objectives**

In order to realise the aims of the research study, multiple hypotheses were formulated and tested. Each of these hypotheses were aimed to compare the two modelling techniques, either in terms of performance or maintenance. The hypotheses are documented below:

### **1.4.1 Load performance**

The mean time required to load historical data into a data vault model is less than the mean time required to load the same data into a dimensional model.

The mean time required to load incremental data into a data vault model is less than the mean time required to load the same data into a dimensional model.

### **1.4.2 Query performance**

The mean query execution time for queries to a data vault model is less than the mean query execution time for similar queries to the dimensional model.

### **1.4.3 Storage space required**

The mean storage space required for data in a data vault model is less than the mean storage space required for the same data in a dimensional model.

### **1.4.4 Flexibility**

The number of existing queries that will be affected after changes/additions to the data vault model have been effected in order to meet changing user requirements is less than the number of existing queries that will be affected on a dimensional model after changes/additions have been made to the model in order to meet the same set of requirements.

## **1.5 Empirical investigation**

An experiment conducted under a positivistic epistemological paradigm was selected as the as the research method for conducting this research study. A justification for this choice is presented in chapter 2. The experiment design and research method, measuring instruments, validity procedures and quality control, proposed methods of data processing, and expected outcomes will be discussed next.

### **1.5.1 Design and method**

A single Third Normal Form (3NF) OLTP source database was used to create two data warehouses (two cases) using dimensional modelling and data vault modelling respectively. In order to prove the hypotheses discussed in section 1.4 above, several measurements were taken on each of the two cases and statistically compared.

With regards to the load performance experiment, a distinction was made between history loads and incremental loads. The time required to load history and incremental data was measured. As regards the query performance hypothesis, queries were divided into different categories according to their complexity. Queries were then written to execute against the dimensional model and the data vault model so as to return equivalent results. Query execution time was also measured. The storage space of each of the data warehouses was measured using different volumes of data in order to test the storage size hypothesis. As regards the flexibility hypothesis, various changes/additions were made to each model in order to meet a set of fictitious, changing user requirements. A count of all the existing queries that were affected by the change was recorded and the counts then compared.

### **1.5.2 Measuring instruments**

Most of the measurements were taken directly from the database management system (DBMS). These include, among others, database file sizes, query execution times. SQL Server 2008 R2 was the DMBS that was selected for the purposes of this experiment. This software suite was selected because it is one of the major databases currently used in practice (Gartner, 2012), and trial versions are readily available at no cost.

### **1.5.3 Validity procedures and quality control**

In order to ensure the validity of the study, the researcher created a research procedure protocol as described in Olivier (2004:75). This protocol was peer-reviewed to ensure maximum validity.

Database theory was used to confirm the measurements. It was anticipated that this would, in turn, increase the confidence in all the measurements.

### **1.5.4 Proposed method of data processing**

All the measurements were analysed using quantitative data analysis methods. The data analysis was conducted in conjunction with the statistical consultation service of the North-West University. The results of the statistical analysis were interpreted by the researcher.

### **1.5.5 Expected outcomes**

The researcher expected the research results to support the load performance, storage requirement and flexibility hypotheses. In other words, it was expected that the data vault model would outperform the dimensional model with regards to load performance, storage requirements and flexibility. However, the researcher had little confidence that the query performance of the dimensional model would be superior to that of the data vault model. This was in view of the fact that the resulting data vault model would produce more tables than the dimensional model. More tables equate to more join operations between tables which, in turn, generally result in slower running queries (Kimball *et al.*, 2008:237). However, there are various design aspects other than the number of join operations that may have an effect on query performance.

Overall the researcher expected the data vault modelling technique to be superior to the dimensional modelling technique in terms of performance and maintenance.

## **1.6 Chapter division**

Chapter 2 details the research methodology that the researcher adopted in conducting this research study. The chapter also discusses the researcher's motivation with regards to his choice of a true experiment as a research method. Chapter 3 contains a literature study on data warehouse modelling, in particular, dimensional modelling and data vault modelling. It also contains a critical evaluation and comparison of the two techniques. Chapter 4 discusses all

aspects related to the experiments, for example, the selection of datasets, measures, and so forth. The actual experiments, research findings and the quantitative analysis of the data are also presented in this chapter. The fifth chapter concludes the research study and contains a summary of the research findings, possible limitations of the study and suggestions for future research.

## **1.7 Summary**

This chapter provided evidence in respect of the need for research that compares the impact of data vault modelling on data warehouse performance and maintenance with the impact of dimensional modelling on data warehouse performance and maintenance. The chapter also provided an overview of the research objectives and the approach that would be adopted to realise these objectives.

# Chapter 2: Research methodology

## 2.1 Introduction

Chapter 1 defined the aim of this research study as a scientific comparison between data warehouse implementations based on dimensional modelling techniques and those based on data vault modelling techniques in terms of performance and maintenance through objective measurements. It was, thus, essential to adopt an appropriate research methodology in order to ensure that the aim of the research study was realised.

This chapter starts with a discussion on the sometimes conflicting terminology used in the research space, and includes descriptions of the following concepts: research paradigm, research method, data collection techniques and research classification.

This is followed by discussion on three different research paradigms, namely, positivism, interpretivism and critical social theory as well as the ontological and epistemological assumptions that underlie each of these research paradigms. A discussion on the hypothetico-deductive method and its relation to positivism and true experiments as well as a detailed discussion on experiments, including Bernard's five-step method for conducting true experiments (Bernard, 2000:105) follows.

Finally, the classification of research data, research methods and qualitative and quantitative research are discussed.

## 2.2 Research terminology

The terminology in research methodology is often used interchangeably in the literature. This section aims to clarify the terminology that will be used in this dissertation. De Villiers (2005:144) describes research paradigms as "the primary philosophical point of departure" in conducting research, while Oates (2006:282) defines a research paradigm as a researcher's worldview which is influenced by both his/her view of the existence of the world and the means by which knowledge may be acquired. On the other hand, Myers (2010) defines the research paradigm as the researcher's "underlying assumptions about what constitutes 'valid' research and which research methods are appropriate". Myers refers to this as the underlying epistemology. In this research study the term 'research paradigm' is used to describe the



paradigm in terms of which research was conducted and which was influenced by the ontological and epistemological assumptions of the researcher. Examples of research paradigms include positivism, interpretivism, and critical social theory (Klein & Myers, 1999:70; Myers, 2010; Oates, 2006:282; Orlikowski & Baroudi, 1991:6; Walsham, 1995:376).

Oates (2006:35) describes a research method as a research strategy and maintains it may be regarded as the overall approach adopted in order to obtain the answer to the research question. Yin (2003:1) uses similar terminology to Oates. Myers (2010) describes a research method as “a strategy of inquiry which moves from the underlying philosophical assumptions to research design and data collection”. This view is shared by Yin and Heald (1975:1). The current study adopts Myers’ view. Examples of research methods include, among others, case studies, action research and surveys (Bakerville, 1999:2; Myers, 2010; Oates, 2006:35; Orlikowski & Baroudi, 1991:4).

Myers (2010) and Oates (2006:36) both describe research techniques as the methods which are used to generate research data, while De Villiers (2005:144) describes research techniques as the “practical means/strategies/techniques/instruments used for data collection”. Examples of research techniques include, among others, interviews, observations, questionnaires (Mingers, 2001:241; Myers, 2010; Oates, 2006:36).

Myers (2010) and Bernard (2000:418) both classify research methods as either qualitative or quantitative, based on the nature of the data that is produced by the data collection techniques comprising the research method. Straub, Gefen and Boudreau (2005) classify a research study as a whole as either qualitative or quantitative, based on the nature of the data that is produced through the research. Both views are accepted and adopted in this study.

Table 2.1 below summarises the terminology used by the various writers discussed above. The table also highlights the terminology and definitions that are used in this study.

**Table 2.1 Summary of research terminology by different writers as well as the terminology and definitions adopted in this study**

|                            | Author                               | Terminology                    | Definition                                                                                                                                                 | Examples                                               |
|----------------------------|--------------------------------------|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| Research paradigm          | Myers, Orlikowski & Baroudi, Walsham | Research epistemology          |                                                                                                                                                            | Positivism, interpretivism and critical social theory  |
|                            | Klein & Myers                        | Research philosophy            |                                                                                                                                                            |                                                        |
|                            | Oates                                | Philosophical paradigm         |                                                                                                                                                            |                                                        |
|                            | This study                           | Research paradigm              | A paradigm under which research is conducted that is influenced by the ontological and epistemological assumption of the researcher                        |                                                        |
| Research method            | Myers, Yin & Heald, Bakerville       | Research method                |                                                                                                                                                            | Survey, case study and experiment, action research     |
|                            | Mingers                              | Research method or techniques  |                                                                                                                                                            |                                                        |
|                            | Oates, Yin                           | Research strategy              |                                                                                                                                                            |                                                        |
|                            | Orlikowski & Baroudi                 | Research design                |                                                                                                                                                            |                                                        |
|                            | This study                           | Research method                | "...a strategy of inquiry which moves from the underlying philosophical assumptions to research design and data collection" (Myers, 2010)                  |                                                        |
| Data collection techniques | Straub <i>et al.</i>                 | Data collection techniques     |                                                                                                                                                            | Interviews, observations, questionnaires and documents |
|                            | Myers, Oates                         | Data generation method         |                                                                                                                                                            |                                                        |
|                            | This study                           | Data collection techniques     | Techniques for collecting empirical data or materials                                                                                                      |                                                        |
| Research classification    | Bernard, Myers                       | Research method classification |                                                                                                                                                            | Quantitative or qualitative                            |
|                            | Oates                                | Data classification            |                                                                                                                                                            |                                                        |
|                            | This study                           | Research classification        | A classification of the research data, the research methods, as well as the research as a whole, based on the nature of the data produced by the research. |                                                        |

The next section will discuss the different research paradigms and indicate which paradigm the researcher subscribes to.

## **2.3 Research paradigms**

A research paradigm is concerned with the construction of valid knowledge about the social and physical reality (Hirschheim & Klein, 1989:1199). There are three important concepts present in this definition, namely, physical and social reality, the construction of knowledge, and ensuring that this knowledge is valid. Ontology is concerned with the existence of reality; epistemology is concerned with the construction of knowledge; and methodology is concerned with the validity of the constructed knowledge through the selection of appropriate research methods (Chua, 1986:604; Hirschheim & Klein, 1989:1201; Orlikowski & Baroudi, 1991:8). Accordingly, in order to offer a complete definition of research paradigms it is essential that each of these concepts be discussed.

Ontology is concerned with the question of whether an objective reality exists (Al-hakim & Carter-Steel, 2009:3). Orlikowski and Baroudi (1991:7) describe ontology as “beliefs [that] have to do with the essence of phenomena under investigation”. This corresponds with Hirschheim’s (1985:3) description of ontology. There are many schools of thought with regards to ontology. However, only a few of them are relevant to this discussion. Realism holds that an objective reality exists independent of the human mind and that it is measurable (Al-hakim & Carter-Steel, 2009:4). Chua (1986:606) explains realism in terms of subject and object. Real objects exist and are independent of the researcher (subject). Knowledge is gained when the subject becomes correctly aware of the properties of the object. Relativism is in sharp contrast to realism (Hirschheim, 1985:13). According to Hirschheim (1985:13), “reality is a subjective construction of the mind” and no objective reality exists. Relativism holds that an “objective reality” may be perceived differently by different subjects, depending on language, religion, culture, etc. Phenomenology attempts to bridge the gap between realism and relativism (Al-hakim & Carter-Steel, 2009:3) with phenomenologists suggesting that objects exist outside the mind, but that human beings may become aware of them only through their minds. Phenomenologists accept the existence of an objective physical reality, but differentiate it from social reality (Al-hakim & Carter-Steel, 2009:4).

Myers (2010) defines epistemology as the assumptions about knowledge, and the way in which ‘valid’ knowledge may be obtained while Hirschheim (1985:1) describes epistemology as the “theory of knowledge; in particular, how we acquire knowledge”. Two distinct concepts arise from these definitions. Firstly, what is knowledge? And, secondly, how valid knowledge can be

obtained? In line with the Greeks, Hirschheim (1985:1) classifies knowledge into one of the two following classes, 1) that which is thought to be true (doxa) and 2) that which is known to be true (episteme). Science is defined as the process of transforming doxa to episteme. This, in turn, implies that knowledge is obtained through science.

Science is always conducted within a specific paradigm (Myers, 2010). Oates (2006:13) defines a paradigm as a shared way of thinking. In other words, a research paradigm is a shared way of thinking among researchers about the way in which knowledge may be acquired. Thus, what constitutes valid knowledge and how it may be acquired depends on the specific research paradigm. With regards to the way in which knowledge may be obtained, research paradigms fall into one of three broad categories; namely, those that believe that knowledge may be obtained only through experience, or intellect, or both (Al-hakim & Carter-Steel, 2009:7).

Methodology is concerned with the research methods and techniques that are appropriate for “gathering valid empirical evidence” (Orlikowski & Baroudi, 1991:8). Whether research methods and techniques may be considered as appropriate depends on the researcher’s paradigm and research methods and techniques that are considered appropriate and that will produce valid results for one paradigm may not be considered appropriate for another paradigm. Accordingly, in the latter instance, the results would not be considered to be valid.

Three research paradigms are prevalent in computer science or information systems research; namely, positivist, interpretive and critical social theory (Al-Hakim & Carter-Steel, 2009:7; Myers, 2010; Orlikowski & Baroudi, 1991:5).

### **2.3.1 Positivism**

Positivism is based on the ontological assumptions of realism and positivists “assume an objective physical and social world that exists independent of humans, and whose nature can be relatively unproblematically apprehended, characterised, and measured” (Orlikowski & Baroudi, 1991:9). Positivist research generally takes objective measurements of reality through observations, and uses these measurements either to support or to disprove a scientific theory (Myers, 2010). Hirschheimdev (1985:3) characterises positivism in terms of five main pillars:

1. *The unity of the scientific method.* De Villiers (2005:1) maintains that positivism equates to the scientific method. The scientific method is the main methodology for positivistic science and the term is sometimes used interchangeably with positivism. On the other hand, unity refers to the fact that the scientific method may be used to acquire

knowledge about a domain of science, either physical or social (Hirschheim, 1985:3). The scientific method is discussed in more detail later in this chapter.

2. *The search for causal relationships.* Positivist research searches for cause-and-effect relationships between the variables under investigation (Hirschheim, 1985:3). Causal relationships are usually stated in the form “when X occurs, then Y happens as well”.
3. *The belief in empiricism.* In terms of empiricism the data gathered for research is believed to be valid only if it is experienced by the senses (Al-Hakim & Carter-Steel, 2009:11).
4. *Science and its processes are value-free.* The moral, religious, social, political, cultural or ideological beliefs of the researcher have no impact on the object and observations under consideration (Hirschheim, 1985:3).
5. In the main, *logic and mathematics constitute the foundation of science.* In view of the fact that the data gathered in positivist research is mostly quantitative, mathematics and logic provide the perfect formal language with which to analyse the data (Hirschheim, 1985:3).

Karl Popper differentiates between scientific theory and myth (Chua, 1986:607). Scientific theories are those which may be empirically falsified whereas myths are “theories” that may never be disproven (Straub *et al.*, 2005). Positivist research theories may only be supported and never proved, while theories are supported as long as their predictions are supported by data.

### **2.3.2 Interpretive research**

Interpretive research is based on the ontological paradigm of phenomenology (Myers, 2010). Myers (2010) defines interpretive research as that which “attempt[s] to understand phenomena through the meanings that people assign to them”. In contrast to positivist research, interpretive research is both subjective and value-based. This is as a result of the fact that the researcher’s value systems may affect the outcome of the research (Orlikowski & Baroudi, 1991:13). Klein and Myers (1999:72) formulated the following set of principles to which to adhere when conducting interpretive research in information systems:

1. *The fundamental principle of the hermeneutic circle.* The hermeneutic circle refers to the process in terms of which the researcher tries to understand the whole by iterating between understanding each of the constituent parts of the whole and their relationships to each other, and then back to understanding the whole. The understanding of the

constituent parts and the whole improves with each iteration of the hermeneutic circle (Klein & Myers, 1999:71).

2. *The principle of contextualisation.* It is essential that interpretive researchers take into account the historical on social context of the research setting. There is, inevitably, a difference in the understanding of different researchers and this difference is caused by the “historical distance” between them (Klein & Myers, 1999:73).
3. *The principle of interaction between researchers and subjects.* Researchers must be aware of the fact that data is generated through social interaction between the researcher and his/her subjects (Klein & Myers, 1999:74).
4. *The principle of abstraction and generalisation.* Interpretive researchers must try to abstract the problem under investigation and generalise the findings. Without this principle the findings of any particular research will not be practically valuable (Klein & Myers, 1999:75).
5. *The principle of dialogical reasoning.* Interpretive researchers realise that every research project starts with a set of preconceptions. However, some of these may assist the researcher in truly understanding the topic under investigation. The researcher must be aware of this and allow the data to change these preconceptions if required (Klein & Myers, 1999:76).
6. *The principle of multiple interpretations.* Interpretive researchers expect to obtain multiple interpretations of social reality from different subjects. Accordingly, they should try to understand the cause of these differences.
7. *The principle of suspicion.* This principle requires a critical evaluation of possible “distortions in the narratives collected from the participants” (Klein & Myers, 1999:77).

According to Klein and Myers (1999:79), all the principles are based essentially on the fundamental principle of the hermeneutic circle. However, these principles are all interdependent. According to the fundamental principle of the hermeneutic circle researchers will move from a focus on each of the principles to a focus on the whole (Klein & Myers, 1999: 84).

### **2.3.3 Critical social research**

Ontologically, critical social research is based on the philosophy that “social reality is historically constituted and, hence, that human beings, organisations, and societies are not confined to existing in a particular state” (Chua, 1986:619). Critical social researchers recognise that, although people are not confined to existing within a particular state, they are often constrained

by social, political or cultural domination (Myers, 2010). The main task of critical social research is that of emancipation (Hirschheim & Klein, 1994:87). This emancipation is achieved through the social critique during which the researcher brings social, political or cultural domination to light (Myers, 2010).

#### **2.3.4 Position of this study**

In view of the fact that the researcher subscribes to the ontological and epistemological views that underlie the positivistic research paradigm, the research in this study will be conducted under the positivistic research paradigm.

### **2.4 Hypothetico-deductive method**

The hypothetico-deductive method, also known and hereafter referred to as the scientific method, is widely acknowledged as the only valid method for producing knowledge under the positivistic research paradigm (Hirschheim, 1985:3; Oates, 2006:127). The scientific method consists of the following six steps, which are depicted below in figure 2.1 (Curiel, 2001:425; Dodig-Crnkovic, 2002:3):

1. The first step involves defining the research question. The research question may be addressed either by existing theories or it may produce new theories.
2. The next step involves formulating a hypothesis as a tentative answer to the research question.
3. Based on existing knowledge and theory, deductions are made that may predict the outcome of the experiment and, therefore, the accuracy of the hypotheses.
4. The fourth step involves carrying out the experiment that will either support or refute the hypothesis. Depending on the results of the experiment at this time, it may be necessary to go back to the second step in order to make adjustments to the hypothesis. In general, steps two, three and four are repeated until the hypothesis is supported by the results of the experiment.
5. A new theory which is supported by scientific evidence may now be proposed. It is essential that this theory, together with details on how the knowledge was produced, be published. During this step the existing theories which were used to make the predictions in the third step may either be confirmed or contradicted. The contradiction of existing theories does not happen often, and is called a scientific revolution.

6. Once the new theory has been published, it will be scrutinised by peers and, through a process of natural selection, either be added to the existing body of knowledge used to make predictions in future experiments, or be discarded by the scientific community.

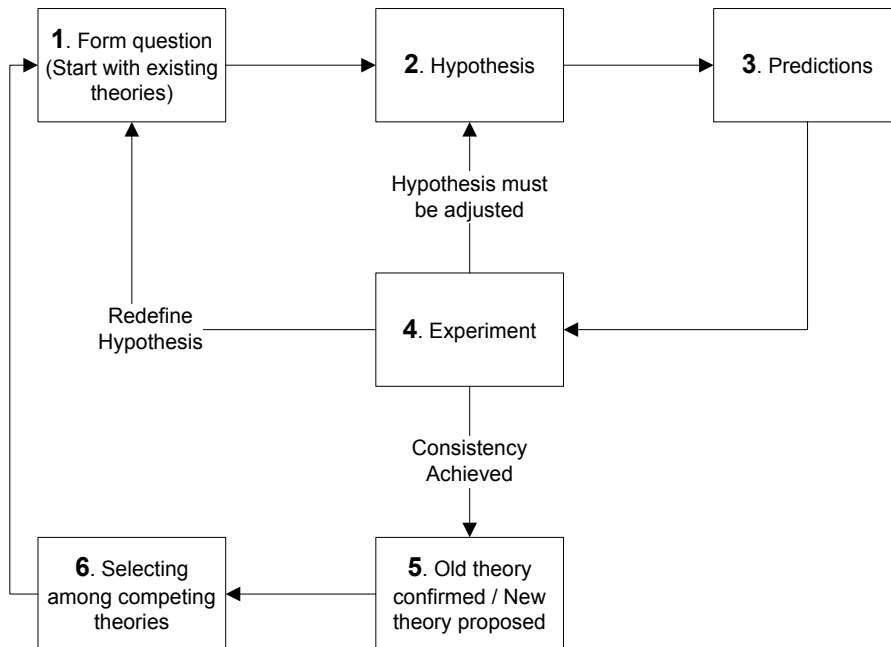


**Figure 2.1 The scientific method (Dodig-Crnkovic, 2002:3)**

The next section will discuss experiments in more detail and will include a discussion on the definition of experiments, experimental design, a five-step method for conducting experiments, hypotheses and validity.

## 2.5 Experiments

According to Oates (2006:127), experiments are “at the heart of the scientific method and positivism”. Oates (2006:127) describes an experiment as “a strategy [research method] that investigates the cause and effect relationships, seeking to prove or disprove a causal link between a factor and an observed outcome”. According to this definition, both the factor and the observed outcome are variables while a causal link between two variables is a relationship in which a change in one variable always causes a change in the other variable (Trochim, 2012). Variables in experiments fall into one of two categories; namely, independent variables or dependent variables. Variables that cause changes to other variables are categorised as



independent variables while variables that change as a result of changes in independent variables are categorised as dependent variables (Oates, 2006:129).

### 2.5.1 Experimental design

Experiments must conform to specific designs. According to Trochim (2012), an experimental design expresses the structure of the experiment. Experimental designs are usually expressed in a formal notation, have a specific name and address the following elements of the experiment:

1. *Observations or measures.* The symbol “O” is used to symbolise observations or measures in the design notation. O may represent either single or complex observations or measurements made during the execution of the experiment. If a distinction between different observations/measurements is required, this distinction may be expressed using subscripts, for example O<sub>1</sub>, O<sub>2</sub>, and so forth (Trochim, 2012).
2. *Groups.* Experiments always involve one or more groups of participants or testing units. Each group has one or more testing units. Each line in the notation denotes a group. For example, a design notation with four lines has four distinct groups of testing units (Trochim, 2012).
3. *Assignment to groups.* Each testing unit is assigned to a specific group in a particular manner. The means of assigning testing units to groups are denoted by a single letter at the beginning of each line/group of the notation. The most popular assignment techniques are random assignment (R), non-equivalent groups (N) or assignment by cut-off (C). Random assignment randomly assigns testing units to a predefined number of groups and, as a result, there is a high probability that the groups are equal. Non-equivalent groups usually make use of some sort of previously defined groups. For example, an experiment may wish to prove the effectiveness of a new educational technique in schools. Thus, instead of randomly assigning the pupils to experimental groups, the researcher may want to keep the pupils in their predefined classes. This design is often termed a field experiment because it takes place in the field (the classroom in this example). In the example cited above, assignment by cut-off may divide the pupils into groups according to their surnames. For example, pupils with surnames starting with A to N would be in one group while pupils with surnames starting with M to Z would be in the second group (Trochim, 2012).
4. *Treatments or interventions.* Treatments or interventions are represented with the symbol “X” in the design notation. Treatments may range from a single intervention to a complex programme of treatments. The absence of an “X” means that the

treatment/interventions are not being applied to the group in question. A group to which no treatment is applied is known as the control group. Treatments are applied to some groups, but not to the control group. However, observations/measurements are taken from all the groups. The purpose of a control group is to ensure that any change in measurement before and after the application of a treatment was, in fact, caused by the treatment, and not by some other uncontrolled factor (Trochim, 2012).

5. *Time*. In the design notation, time moves from left to right with elements that are listed to the left of the notation occurring before elements that are listed to the right. Elements in different groups (lines of the notation) that are underneath each other occur at the same time (Trochim, 2012).

As an example of experimental design notation, consider figure 2.2 below. This design is termed a randomised, pretest–posttest with control group, experimental design. The two lines in the notation indicate two distinct groups. The “R” symbol at the beginning of each line indicates that the testing units of these groups were randomly assigned to the groups. The absence of the “X” in the second line indicates that the treatment was not applied to this group. This group is, therefore, called the control group. The timeline, read from left to right, indicates that the groups were first assigned, a single measurement was taken for each group (denoted by “O<sub>1</sub>”), a treatment was applied to one group and, lastly, two measurements/observations were taken (denoted by “O<sub>1,2</sub>”).

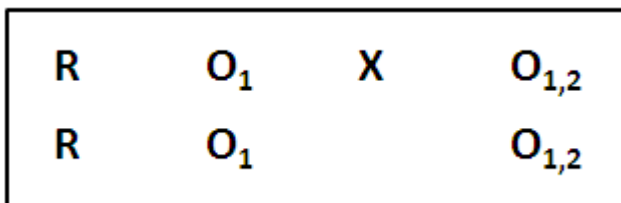


**Figure 2.2 Experimental design notation**

Experimental designs are divided into the following three distinct types: 1) true or randomised experiments, 2) quasi- or field experiments, or 3) non-experiments (Oates, 2006:128; Trochim, 2012). The main difference between field experiments and the other types of experiment is that field experiments are conducted in naturally occurring environments whereas the other types are conducted in laboratories or controlled environments (Thomas, 1996:115). Trochim (2012) depicts a decision tree that assists the researcher in selecting the correct type of experimental design. This tree is presented in figure 2.3 below.

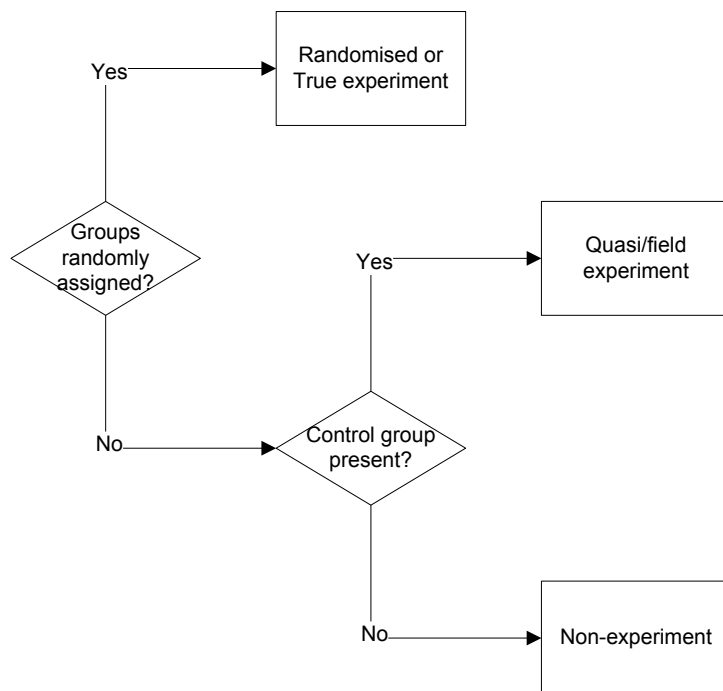


**Figure 2.3 Design type decision tree (Trochim, 2012)**

Despite the fact that the experimental groups in this dissertation were not randomly assigned, the researcher will argue in chapter 4 that the manner in which the groups were constituted caused the groups to be equal – the same effect that random assignment seeks to achieve. Accordingly, the researcher conducted a true experiment.

### **2.5.2 The five-step method for true experiments**

Bernard (2000:105) describes five typical steps to follow when conducting true experiments:

1. Formulate the hypothesis.
2. Randomly assign testing units to groups.
3. Measure the dependent variables in all groups. This is called the pre-test and happens only in experimental designs in which a pre-test is required (Trochim, 2012).
4. Apply the treatment or intervention to all groups except the control group.
5. Measure the dependent variables in all groups. This is called the post-test (Trochim, 2012).

### 2.5.3 Hypothesis

As mentioned in the previous sections, the first step in conducting true experiments is to formulate the hypothesis (Bernard, 2000:105). The purpose of an experiment is either to prove or disprove causal relationships between two or more variables. This is done by either proving or falsifying a hypothesis (Oates, 2006:128). A hypothesis is a statement that has not yet been scientifically proven. However, it is possible to gather empirical data that will clearly support or refute the hypothesis (Oates, 2006:128; Trochim, 2012). As an example, consider the following hypothesis:

*H<sub>A</sub>: A change in the underlying data model of a data warehouse from a dimensional to a data vault model will cause a significant decrease in query performance.*

In the example above the hypothesis predicts the effect that the change in the independent variable, namely, the data model, will have on the dependent variable, namely, query performance. In order to prove the above hypothesis to be correct, it is essential that all other outcomes be proven false. Therefore, if the experiment is able to prove that the opposite of the hypothesis above is false, this, in turn, would imply that the above hypothesis is true. The opposite of the hypothesis cited above is referred to as the null hypothesis, denoted by H<sub>0</sub>. The hypothesis that a researcher aims to prove through an experiment is referred to as the alternative hypothesis, denoted by H<sub>A</sub>. The matching null hypothesis for the alternative hypothesis above would be as follows:

*H<sub>0</sub>: A change in the underlying data model of a data warehouse from a dimensional to a data vault model will either have no impact or cause a significant increase in query performance.*

### 2.5.4 Validity

Validity is an important concept in all experiments. Trochim (2012) defines validity as “the best available approximation to the truth of a given proposition, inference, or conclusion”. Thus, this definition implies that any propositions, inferences or conclusions obtained from an experiment that has poor validity may not be trusted. Research that suffers from poor validity will not pass a peer review and will not become part of the existing body of knowledge (Dodig-Crnkovic, 2002:3). Validity may be subdivided into the following four types (Oates, 2006:131; Trochim, 2012):

- *Conclusion validity.* Trochim (2012) define conclusion validity as “the degree to which conclusions we reach about relationships in our data are reasonable”.
- *Internal validity.* Assuming that the research has conclusion validity, that is, there is a relationship between the independent and dependent variable, internal validity is concerned with whether this relationship is causal or not (Oates, 2006:131; Trochim, 2012). In other words, is it possible for a researcher to conclude that a change in the independent variable is responsible for a change in the dependent variable?
- *Construct validity.* In order to understand construct validity, it is necessary to understand the two realms that are involved in research, namely, theory and observation. In the realm of theory, the researcher formulates a theory about both cause, termed the cause construct, and effect, termed the effect construct. When executing the research, the researcher moves from the realm of theory to the realm of observation. This move from theory to observation translates a cause construct into a treatment/program. Similarly, an effect construct is translated into observations. Trochim (2012) refers to this translation as operationalising the constructs. Given conclusion and internal validity, construct validity is concerned with how credible the operationalisation of the constructs are (Trochim, 2012).
- *External validity.* External validity is concerned with the generalisability of the conclusions of a research study (Oates, 2006:132; Trochim, 2012). If an experiment has conclusion validity, internal validity, and construct validity, is it possible to generalise the results of the research to other instances?

Poor conclusion validity may result in a researcher either missing relationships that exist, or “seeing” relationships that do not exist while poor internal validity may cause a researcher to conclude incorrectly either that a relationship is causal or else he/she may miss causal relationships. Poor construct validity may lead to incorrect variables being measured, or a treatment being applied incorrectly while poor external validity may mean that the conclusions of the research are applicable only to the scenario under observation, and that it is not possible to generalise these conclusions. Clearly, poor validity does not lead to valid research.

## 2.6 Qualitative or quantitative research?

Research itself, research methods and research data may be classified as either qualitative or quantitative, with the nature of the data determining whether it is classified as quantitative or qualitative. Quantitative data is always numeric while qualitative data is non-numeric, for

example text (Oates, 2006:266; Myers, 2010). If a research method and the accompanying data collection technique produce qualitative data, then the research method is classified as qualitative. The same is true for research methods that produce quantitative data. Research as a whole may also be classified as either qualitative or quantitative in a similar manner. Consider the following two examples: In a research project that uses an interpretive case study as the research method and interviews as the data collection method and that produces qualitative data, the research project as a whole, as well as the research method and research data, may be classified as qualitative. Similarly, in a research project that uses a positivistic experiment as the research method and objective measurements as the data collection technique and that produces quantitative data, the research project as a whole, as well as the research method and the research data, may be classified as quantitative.

It should be noted that, despite the fact that data is either exclusively qualitative or quantitative, research methods may be classified as either qualitative or quantitative. For example, an experiment may produce either qualitative or quantitative data. The class (qualitative or quantitative) of data produced by a research method is often determined by the research paradigm selected. Continuing with the example cited above, experiments conducted under a positivistic paradigm will produce quantitative data while an experiment conducted under an interpretive paradigm can produce either qualitative or quantitative data. According to Myers (2010), qualitative research can be conducted under each of the research paradigms. Accordingly, qualitative research may be either positivistic, interpretive, or social critical. According to Straub *et al.* (2005), quantitative research may be conducted only under the positivistic research paradigm. The relationship between the various research paradigms and the classification of the data produced under each are depicted in figure 2.4 below.

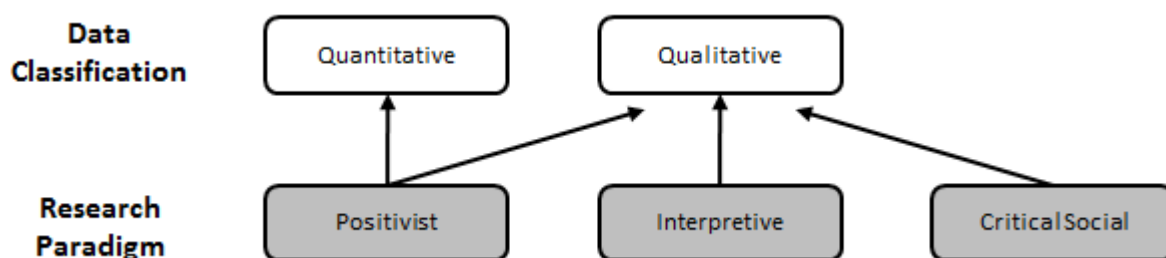


**Figure 2.4 Research paradigm and classification of data (Straub *et al.*, 2005)**

The true experiment that was used in this study made use of objective measurement as the data collection technique. These measurements produced quantitative data. Accordingly, the

experiment data, research method, and the research as a whole may be classified as quantitative.

## **2.7 Summary**

The aim of this research study is to scientifically compare data warehouse implementations based on dimensional modelling techniques with those based on data vault modelling techniques in terms of performance and maintenance through objective measurements. The aim of this chapter was to ensure that appropriate research methods and techniques were selected in order to ensure the realisation of the research aims and also that the study would construct valid knowledge that would be accepted by the scientific community.

Various research methods were introduced. Yin's process of elimination was used and a true experiment was selected as the appropriate research method for this study (Yin, 2003:5). The selection of an experiment as the best possible research method was confirmed by Oates' definition of experiments (Oates, 2006:127). Experiments as a research method were described in detail while the issue of validity was also discussed. The validity of experiments ensures that they produce valid research that will be accepted by the scientific community.

The underlying ontological and epistemological views of positivism, interpretivism and critical social theory as research methods were discussed. In view of the fact that the researcher subscribes to the positivistic research paradigm and the research produced quantitative data, this research as a whole may be classified as quantitative positivistic research (QPR) (Straub *et al.*, 2005).

In the next chapter, a detailed literature study is conducted on both dimensional modelling and data vault modelling; the two data modelling technique under investigation in this study.

# Chapter 3: Data warehouse modelling

## 3.1 Introduction

In this chapter the subject of data warehouse modelling is discussed. The chapter serves as an introduction to data warehousing and will describe the background that is required for the remaining chapters. However, the chapter is not intended as a complete description of the subject area.

The chapter starts by defining a data warehouse, data warehousing, business intelligence and data warehouse modelling techniques. An example of a DVD retailer is then described. This example is used throughout the remainder of the chapter to assist with the explanation of the different modelling techniques.

Although various data warehouse modelling techniques do exist (Raisinghani, 2000:73), two of these only form part of this study; namely, dimensional modelling and data vault modelling. Dimensional modelling is discussed in detail. The discussion includes fundamental concepts such as facts, dimensions and hierarchies. A few advanced dimensional modelling concepts are discussed, including snowflaking, slowly changing dimensions, many-to-many dimension relationships and ragged hierarchies.

The discussion on dimensional modelling will be followed by a section on data vault modelling, a new modelling technique pioneered by Dan E Linstedt in 2002 (Linstedt, 2002). This technique requires a paradigm shift for traditional dimensional modellers. The DVD retail example will assist with this shift by providing comprehensive examples. Data vault modelling topics include hub-, link- and satellite entities while advanced topics on data vault modelling are also discussed. The section concludes with the description of a seven-step process to data vault modelling.

A theoretical comparison between dimensional modelling and data vault modelling follows, with similarities and differences between the two modelling techniques being discussed. The chapter concludes by trying to find a correlation between the success of a business intelligence/data warehousing project, and the data modelling technique implemented.



## 3.2 Definitions

W.H. Inmon is universally accepted as the father of data warehousing (Drewek, 2005). Inmon, Strauss, and Neushloss (2008:7) define a data warehouse as a subject-oriented, integrated, non-volatile, time-variant collection of data with the ultimate goal of decision support. Each of the keywords in his definition can be further defined:

- *Subject oriented.* Data warehouse analyses occur in terms of subjects as opposed to an organisation's functions or processes. Examples of subjects include customer, products and time.
- *Integrated.* The data of the data warehouse is usually sourced from multiple sources with the data from these sources having to be integrated into the warehouse as a coherent whole.
- *Non-volatile.* Data is only added to the warehouse; never deleted. This, in turn, provides the organisation in question with a consistent view of its data.
- *Time variant.* Any piece of information from the data warehouse is always seen within a specific time context.

The definition cited above has been widely accepted by the data warehousing community.

Ralph Kimball is known as the creator of the data mart (Drewek, 2005). Kimball defined a data mart as "the restriction of the data warehouse to a single business process or to a group of related business processes targeted towards a particular business group" (Kimball *et al.*, 1998:18). He further defines a data warehouse as the union of all the constituent data marts. However, this definition makes sense only when a data mart is defined in more detail, and not in terms of the data warehouse itself. This issue will be discussed in greater detail later in the chapter.

Kimball then redefined the data warehouse, which he termed the enterprise data warehouse (EDW), as "the queryable data in your DW/BI system" (Kimball *et al.*, 2008:602). According to Kimball *et al.* (2008:599), the term "data mart" had been marginalised by other practitioners in the field and, thus, he intentionally steered away from this term in his later definition of a data warehouse. However, the researcher finds Kimball's 2008 definition vague and prefers his initial definition of a data warehouse in terms of a data mart.

The researcher would like to make a distinction between the following terms that are used in this research study as they are sometimes used interchangeably in the literature:

- business intelligence
- data warehouse
- data warehousing
- data warehouse modelling.

In an interview with Howard Dresner from the Gartner Group, Dresner described “business intelligence” as an umbrella term which is used to describe “concepts and methods to improve business decision making by using fact-based support systems” (quoted by Baum, 2001:26). In this study, a data warehouse is viewed as such a fact-based system. Goede (2005:132) describes the data warehouse as the noun, and data warehousing as the process of creating or maintaining the data warehouse. Data warehouse modelling refers to the data modelling technique which is used to represent/describe the data in the data warehouse. It also determines the physical implementation of the warehouse.

### **3.3 Data warehouse modelling techniques**

Several data warehouse architectures exist (Eckerson, 2005). Each of these is centred and focuses on one or more data warehouse modelling techniques. Accordingly, this section starts with a discussion on two data warehouse modelling techniques, namely, dimensional modelling and data vault modelling.

The reader should note that there are several other data warehouse modelling techniques other than dimensional modelling and data vault modelling that exist. Examples include entity relationship third normal form, anchor modelling and others (Raisinghani, 2000:71). However, these techniques are outside the scope of this research study, as the goal of this study is to compare data warehouse implementations based on dimensional modelling techniques with those based on data vault modelling techniques in terms of performance and maintenance.

### **3.4 Running example**

A running example is used to explain each of the different data warehouse modelling techniques. The example that is used is that of a DVD retailer. This section explains the DVD

retailer example in detail, while also describing an entity relationship model (ERD) and sample data.

The DVD retailer, which is known as Mr DVD, sells one product only, DVDs. Mr DVD is owned and managed by Joe Lennon. Employees of Mr DVD sell to registered customers only. Upon registration, customer information such as address, telephone number, and so forth are captured. The retailer also store detailed information on each DVD title (hereafter referred to as titles), including release dates, the production company that produced the film, starring and supporting actors, etc. A recursive/hierarchical relationship exists between the employees and managers at Mr DVD. This relationship is illustrated in figure 3.1 below.

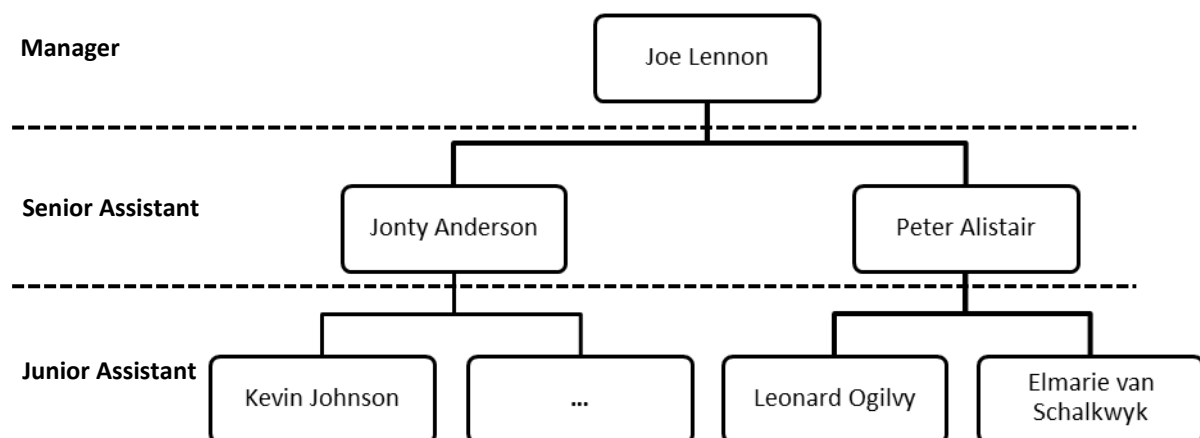


**Figure 3.1 Example of employee hierarchy**

Mr DVD uses a point-of-sale software application known as DVD POS. The ERD from the DVD POS application is depicted in figure 3.2 below.

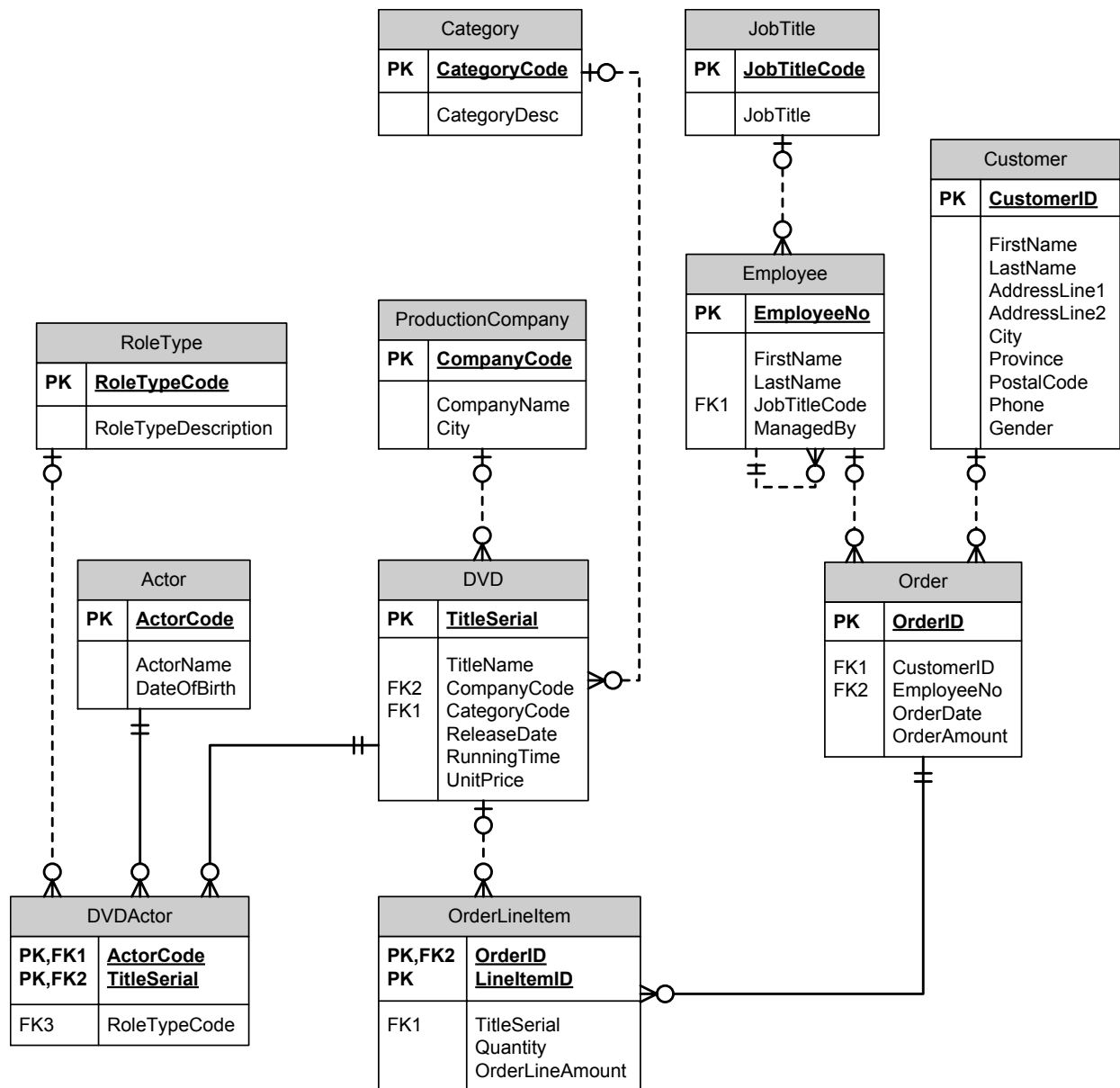


**Figure 3.2 Mr DVD entity relationship diagram**

Sample data for each of the tables in figure 3.2 above is presented in table 3.1 to table 3.11 below.

**Table 3.1 Mr DVD sample data (Category)**

| CategoryCode | CategoryDesc |
|--------------|--------------|
| ACT          | Action       |
| ADV          | Adventure    |
| ANI          | Animation    |
| COM          | Comedy       |
| DRA          | Drama        |
| THR          | Thriller     |

**Table 3.2 Mr DVD sample data (JobTitle)**

| JobTitleCode | JobTitle         |
|--------------|------------------|
| J            | Junior Assistant |
| M            | Store Manager    |
| S            | Senior Assistant |

**Table 3.3 Mr DVD sample data (RoleType)**

| RoleTypeCode | RoleTypeDescription |
|--------------|---------------------|
| L            | Leading             |
| S            | Supporting          |
| V            | Voice               |

**Table 3.4 Mr DVD sample data (Actor)**

| ActorCode | ActorName          | DateOfBirth |
|-----------|--------------------|-------------|
| ALLK0     | Karen Allen        | 1951/10/05  |
| BRAA0     | Andre Braugher     | 1962/07/01  |
| DIAC0     | Cameron Diaz       | 1972/08/30  |
| FORH0     | Harrison Ford      | 1942/07/13  |
| FRAB0     | Brendan Fraser     | 1968/12/03  |
| FREM0     | Morgan Freeman     | 1937/06/01  |
| FREP0     | Paul Freeman       | 1943/01/18  |
| JOLA0     | Angelina Jolie     | 1975/06/04  |
| KUTA0     | Ashton Kutcher     | 1978/02/07  |
| LITJ0     | John Lithgow       | 1945/10/19  |
| MCAJ0     | James McAvoy       | 1979/04/21  |
| MURE0     | Eddie Murphy       | 1961/04/03  |
| MYEM0     | Mike Myers         | 1963/05/25  |
| PLAO0     | Oliver Platt       | 1960/01/12  |
| RUSK0     | Keri Russel        | 1976/03/23  |
| STAS0     | Sylvester Stallone | 1946/07/06  |
| YOUB0     | Burt Young         | 1940/04/30  |

**Table 3.5 Mr DVD sample data (ProductionCompany)**

| CompanyCode | CompanyName        | City        |
|-------------|--------------------|-------------|
| 20T         | 20th Century Fox   | Los Angeles |
| COL         | Columbia Pictures  | Culver City |
| DRE         | DreamWorks         | Glendale    |
| MGM         | MGM Studios        | Los Angeles |
| PAR         | Paramount Pictures | Hollywood   |
| SON         | Sony Pictures      | Culver City |
| UNI         | Universal Studios  | Los Angeles |

**Table 3.6 Mr DVD sample data (Employee)**

| EmployeeNo | FirstName | LastName      | JobTitleCode | ManagedBy |
|------------|-----------|---------------|--------------|-----------|
| AndeJ04    | Jonty     | Anderson      | S            | LennJ01   |
| AlisP02    | Peter     | Alistair      | S            | LennJ01   |
| OgilL08    | Leonard   | Ogilvy        | J            | AlisP02   |
| LennJ01    | Joe       | Lennon        | M            | NULL      |
| JohnK15    | Kevin     | Johnson       | J            | AndeJ04   |
| RobbS14    | Stephen   | Robbie        | J            | AndeJ04   |
| VanSE09    | Elmarie   | Van Schalkwyk | J            | AlisP02   |
| VanSZ12    | Zané      | Van Schalkwyk | J            | AndeJ04   |

**Table 3.7 Mr DVD sample data (DVDActor)**

| ActorCode | TitleSerial | RoleTypeCode |
|-----------|-------------|--------------|
| LITJ0     | CLIFFHAN00  | S            |
| STAS0     | CLIFFHAN00  | L            |
| MURE0     | DRDOLITT00  | L            |
| PLAO0     | DRDOLITT00  | S            |
| FORH0     | EXTRAORD00  | L            |
| FRAB0     | EXTRAORD00  | L            |
| RUSK0     | EXTRAORD00  | L            |
| ALLK0     | INDIANAJ00  | S            |
| FORH0     | INDIANAJ00  | L            |
| FREP0     | INDIANAJ00  | S            |
| STAS0     | ROCKY00000  | L            |
| YOUB0     | ROCKY00000  | S            |
| BRAA0     | SALT000000  | S            |
| JOLA0     | SALT000000  | L            |
| DIAC0     | SHREK00000  | V            |
| LITJ0     | SHREK00000  | V            |
| MURE0     | SHREK00000  | V            |
| MYEM0     | SHREK00000  | V            |
| FREM0     | WANTED0000  | L            |
| JOLA0     | WANTED0000  | L            |
| MCAJ0     | WANTED0000  | S            |
| DIAC0     | WHATHAPP00  | L            |
| KUTA0     | WHATHAPP00  | L            |



**Table 3.8 Mr DVD sample data (Customer)**

| CustomerID    | FirstName | LastName    | AddressLine1         | AddressLine2    | City        | Province | PostalCode | Phone      | Gender |
|---------------|-----------|-------------|----------------------|-----------------|-------------|----------|------------|------------|--------|
| 6908086310075 | Johann    | Kruger      | 436 Aureole Avenue   | NULL            | Midrand     | Gauteng  | 2635       | 0843650136 | M      |
| 7102215895085 | Van Zyl   | Kruger      | 23 Duncan Street     | NULL            | Midrand     | Gauteng  | 2635       | 0836350698 | M      |
| 7406286136008 | James     | Green       | 17 Almeisan          | Rough Street    | Randburg    | Gauteng  | 2045       | 0797213985 | M      |
| 7809184852099 | Elmien    | Geel        | 139 Sibaya Sands     | Straight Street | Roodepoort  | Gauteng  | 1724       | 0723600201 | F      |
| 7905165036035 | Kevin     | Venter      | 188 Ruben Street     | NULL            | Alberton    | Gauteng  | 1983       | 0846564138 | M      |
| 7905255005089 | Spike     | Moony       | 12 Montego Blue      | Kelly Avenue    | Krugersdorp | Gauteng  | 1985       | 0836982401 | M      |
| 8003203652098 | Yolandi   | Langenhoven | 66 Victorian Heights | 2 Anvil Street  | Fourways    | Gauteng  | 1885       | 0825689340 | F      |
| 8011145136078 | Pieter    | Erwee       | 75a Tulbach Road     | NULL            | Pretoria    | Gauteng  | 1368       | 0832992791 | M      |

**Table 3.9 Mr DVD sample data (DVD)**

| TitleSerial | TitleName                      | CompanyCode | CategoryCode | ReleaseDate | RunningTime | UnitPrice |
|-------------|--------------------------------|-------------|--------------|-------------|-------------|-----------|
| CLIFFHAN00  | Cliffhanger                    | SON         | ACT          | 1997/11/26  | 113         | 160.95    |
| DRDOLITT00  | Dr. Dolittle                   | 20T         | COM          | 1999/08/03  | 85          | 180.85    |
| EXTRAORD00  | Extra Ordinary Measures        | SON         | DRA          | 2010/01/22  | 106         | 210.95    |
| INDIANAJ00  | Indiana Jones and the Lost Ark | PAR         | ADV          | 2008/05/13  | 115         | 160.75    |
| ROCKY000000 | Rocky                          | MGM         | ACT          | 2005/02/08  | 119         | 160.95    |
| SALT000000  | Salt                           | COL         | THR          | 2010/07/23  | 103         | 230.95    |
| SHREK00000  | Shrek                          | DRE         | ANI          | 2010/04/26  | 90          | 130.55    |
| WANTED0000  | Wanted                         | UNI         | ACT          | 2008/06/27  | 110         | 170.95    |
| WHATHAPP00  | What Happens in Vegas          | 20T         | COM          | 2008/08/26  | 99          | 210.85    |

**Table 3.10 Mr DVD sample data (Order)**

| OrderID   | CustomerID    | EmployeeNo | OrderDate  | OrderAmount |
|-----------|---------------|------------|------------|-------------|
| MRDVD2036 | 7102215895085 | AlisP02    | 2010/02/12 | 401.90      |
| MRDVD2037 | 7905165036035 | OgilL08    | 2010/02/16 | 321.50      |
| MRDVD2038 | 8003203652098 | VanSZ12    | 2010/05/25 | 341.40      |
| MRDVD2039 | 6908086310075 | VanSE09    | 2010/06/13 | 371.80      |
| MRDVD2044 | 7905255005089 | AndeJ04    | 2010/08/25 | 461.90      |
| MRDVD2045 | 7905165036035 | VanSZ12    | 2010/09/08 | 502.75      |
| MRDVD2051 | 6908086310075 | AndeJ04    | 2010/09/22 | 170.95      |
| MRDVD2052 | 6908086310075 | VanSZ12    | 2010/09/22 | 381.90      |

**Table 3.11 Mr DVD sample data (OrderLineItem)**

| OrderID   | LineItemID | TitleSerial | Quantity | OrderLineAmount |
|-----------|------------|-------------|----------|-----------------|
| MRDVD2036 | 1          | SALT000000  | 1        | 230.95          |
| MRDVD2036 | 2          | WANTED0000  | 1        | 170.95          |
| MRDVD2037 | 1          | INDIANAJ00  | 2        | 321.50          |
| MRDVD2038 | 1          | WHATHAPP00  | 1        | 210.85          |
| MRDVD2038 | 2          | SHREK00000  | 1        | 130.55          |
| MRDVD2039 | 1          | CLIFFHAN00  | 1        | 160.95          |
| MRDVD2039 | 2          | WHATHAPP00  | 1        | 210.85          |
| MRDVD2044 | 1          | SALT000000  | 2        | 461.90          |
| MRDVD2045 | 1          | CLIFFHAN00  | 1        | 160.95          |
| MRDVD2045 | 2          | DRDOLITT00  | 1        | 180.85          |
| MRDVD2045 | 3          | ROCKY00000  | 1        | 160.95          |
| MRDVD2051 | 1          | WANTED0000  | 1        | 170.95          |
| MRDVD2052 | 1          | EXTRAORD00  | 1        | 210.95          |
| MRDVD2052 | 2          | WANTED0000  | 1        | 170.95          |

This example is used to illustrate the different data warehouse modelling concepts and techniques throughout the remainder of this chapter. For each modelling technique, the relevant data model and example data are provided as this will assist the reader in coming to a better understanding of the concepts to follow.

### 3.5 Dimensional modelling

Dimensional modelling refers to a logical data warehouse modelling technique which was pioneered by Ralph Kimball in the late 1980s (Kimball *et al.*, 2008:1). It was deemed to be an improvement on 3NF models in many ways with the key selling points being ease of

understanding and less joins between relational tables that, in turn, resulted in improved query performance (Kimball *et al.*, 2008:237).

The dimensional model for Mr DVD is presented in figure 3.3 below. This model is used to explain the concepts mentioned in the dimensional modelling section. Sample data for the tables is depicted in table 3.12 to table 3.20 below. Note that the Dim\_Date figure contains a subset of data used in the example only while the other tables contain complete sets.

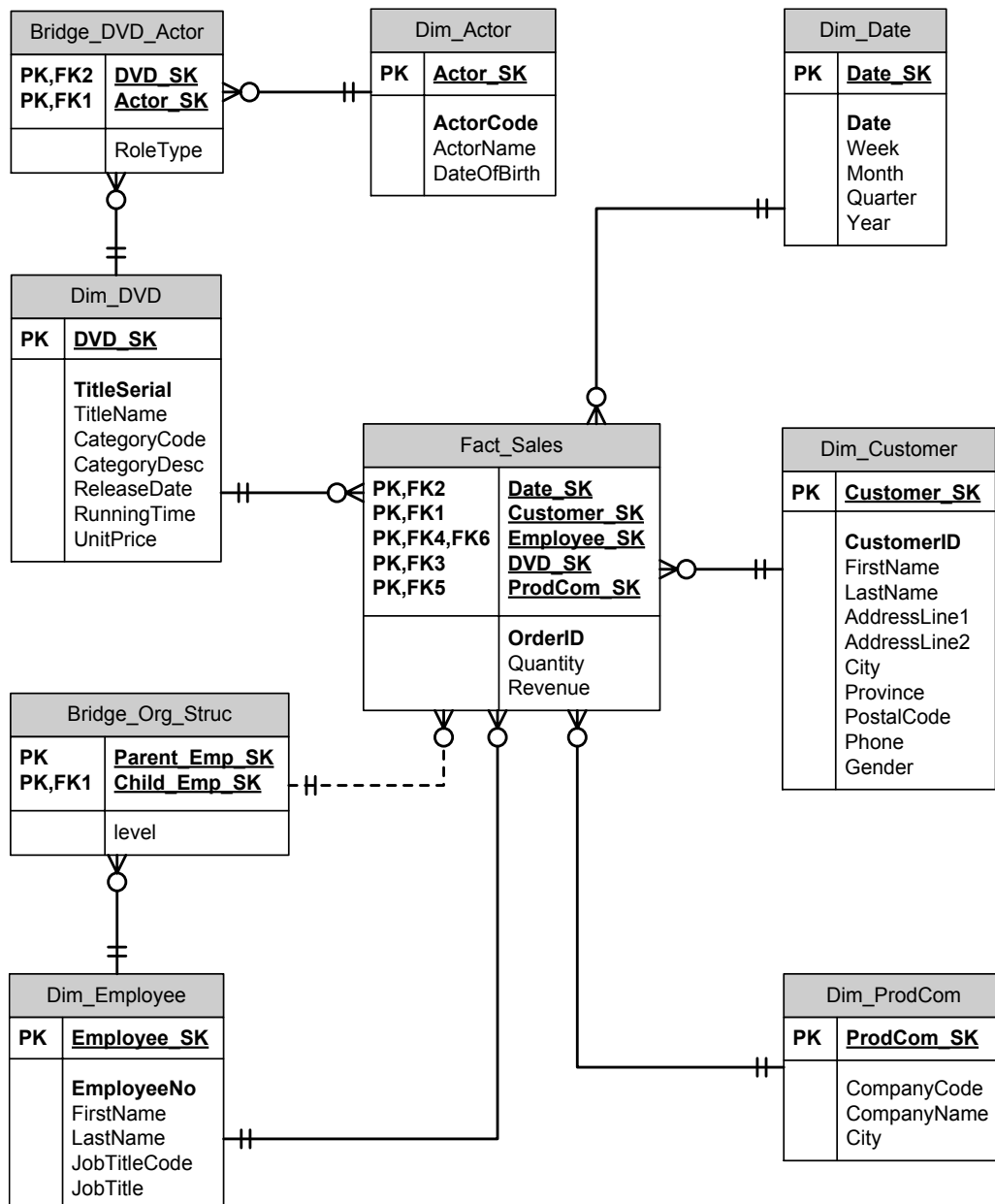


Figure 3.3 Dimensional data model for Mr DVD example

**Table 3.12 Mr DVD sample (Dim\_Customer)**

| Customer_SK | CustomerID    | FirstName | LastName    | AddressLine1         | AddressLine2    | City        | Province | PostalCode | Phone      | Gender |
|-------------|---------------|-----------|-------------|----------------------|-----------------|-------------|----------|------------|------------|--------|
| 46          | 6908086310075 | Johann    | Kruger      | 436 Aureole Avenue   | NULL            | Midrand     | Gauteng  | 2635       | 0843650136 | M      |
| 47          | 7102215895085 | Van Zyl   | Kruger      | 23 Duncan Street     | NULL            | Midrand     | Gauteng  | 2635       | 0836350698 | M      |
| 48          | 7406286136008 | James     | Green       | 17 Almeisan          | Rough Street    | Randburg    | Gauteng  | 2045       | 0797213985 | M      |
| 49          | 7809184852099 | Elmien    | Geel        | 139 Sibaya Sands     | Straight Street | Roodepoort  | Gauteng  | 1724       | 0723600201 | F      |
| 50          | 7905165036035 | Kevin     | Venter      | 188 Ruben Street     | NULL            | Alberton    | Gauteng  | 1983       | 0846564138 | M      |
| 51          | 7905255005089 | Spike     | Moony       | 12 Montego Blue      | Kelly Avenue    | Krugersdorp | Gauteng  | 1985       | 0836982401 | M      |
| 52          | 8003203652098 | Yolandi   | Langenhoven | 66 Victorian Heights | 2 Anvil Street  | Fourways    | Gauteng  | 1885       | 0825689340 | F      |
| 53          | 8011145136078 | Pieter    | Erwee       | 75a Tulbach Road     | NULL            | Pretoria    | Gauteng  | 1368       | 0832992791 | M      |

**Table 3.13 Mr DVD sample data (Dim\_DVD)**

| DVD_SK | TitleSerial | TitleName                      | CategoryCode | CategoryDesc | ReleaseDate | RunningTime | UnitPrice |
|--------|-------------|--------------------------------|--------------|--------------|-------------|-------------|-----------|
| 64     | CLIFFHAN00  | Cliffhanger                    | ACT          | Action       | 1997/11/26  | 113         | 160.95    |
| 65     | DRDOLITT00  | Dr. Dolittle                   | COM          | Comedy       | 1999/08/03  | 85          | 180.85    |
| 66     | EXTRAORD00  | Extra Ordinary Measures        | DRA          | Drama        | 2010/01/22  | 106         | 210.95    |
| 67     | INDIANAJ00  | Indiana Jones and the Lost Ark | ADV          | Adventure    | 2008/05/13  | 115         | 160.75    |
| 68     | ROCKY00000  | Rocky                          | ACT          | Action       | 2005/02/08  | 119         | 160.95    |
| 69     | SALT000000  | Salt                           | THR          | Thriller     | 2010/07/23  | 103         | 230.95    |
| 70     | SHREK00000  | Shrek                          | ANI          | Animation    | 2010/04/26  | 90          | 130.55    |
| 71     | WANTED0000  | Wanted                         | ACT          | Action       | 2008/06/27  | 110         | 170.95    |
| 72     | WHATHAPP00  | What Happens in Vegas          | COM          | Comedy       | 2008/08/26  | 99          | 210.85    |

**Table 3.14 Mr DVD sample data (Bridge\_DVD\_Actor)**

| DVD_SK | Actor_SK | RoleType   |
|--------|----------|------------|
| 64     | 15       | Supporting |
| 64     | 21       | Leading    |
| 65     | 17       | Leading    |
| 65     | 19       | Supporting |
| 66     | 9        | Leading    |
| 66     | 10       | Leading    |
| 66     | 20       | Leading    |
| 67     | 6        | Supporting |
| 67     | 9        | Leading    |
| 67     | 12       | Supporting |
| 68     | 21       | Leading    |
| 68     | 22       | Supporting |
| 69     | 7        | Supporting |
| 69     | 13       | Leading    |
| 70     | 8        | Voice      |
| 70     | 15       | Voice      |
| 70     | 17       | Voice      |
| 70     | 18       | Voice      |
| 71     | 11       | Leading    |
| 71     | 13       | Leading    |
| 71     | 16       | Supporting |
| 72     | 8        | Leading    |
| 72     | 14       | Leading    |

**Table 3.15 Mr DVD sample data (Bridge\_Org\_Struc)**

| ParentEmpSK | ChildEmpSK | Level |
|-------------|------------|-------|
| 34          | 34         | 0     |
| 34          | 38         | 1     |
| 34          | 39         | 1     |
| 34          | 41         | 1     |
| 35          | 35         | 0     |
| 35          | 36         | 1     |
| 35          | 40         | 1     |
| 36          | 36         | 0     |
| 37          | 34         | 1     |
| 37          | 35         | 1     |
| 37          | 36         | 2     |
| 37          | 37         | 0     |
| 37          | 38         | 2     |
| 37          | 39         | 2     |
| 37          | 40         | 2     |
| 37          | 41         | 2     |
| 38          | 38         | 0     |
| 39          | 39         | 0     |
| 40          | 40         | 0     |
| 41          | 41         | 0     |

**Table 3.16 Mr DVD sample data (Dim\_Employee)**

| Employee_SK | EmployeeNo | FirstName | LastName      | JobTitleCode | JobTitle         |
|-------------|------------|-----------|---------------|--------------|------------------|
| 34          | AndeJ04    | Jonty     | Anderson      | S            | Senior Assistant |
| 35          | AlisP02    | Peter     | Alistair      | S            | Senior Assistant |
| 36          | OgilL08    | Leonard   | Ogilvy        | J            | Junior Assistant |
| 37          | LennJ01    | Joe       | Lennon        | M            | Store Manager    |
| 38          | JohnK15    | Kevin     | Johnson       | J            | Junior Assistant |
| 39          | RobbS14    | Stephen   | Robbie        | J            | Junior Assistant |
| 40          | VanSE09    | Elmarie   | Van Schalkwyk | J            | Junior Assistant |
| 41          | VanSZ12    | Zané      | Van Schalkwyk | J            | Junior Assistant |

**Table 3.17 Mr DVD sample data (Dim\_Date)**

| Date_SK  | Date       | Week | Month   | Quarter | Year |
|----------|------------|------|---------|---------|------|
| 20100101 | 2010/01/01 | 1    | January | Q1 2010 | 2010 |
| 20100102 | 2010/01/02 | 1    | January | Q1 2010 | 2010 |
| 20100103 | 2010/01/03 | 2    | January | Q1 2010 | 2010 |
| 20100104 | 2010/01/04 | 2    | January | Q1 2010 | 2010 |
| 20100105 | 2010/01/05 | 2    | January | Q1 2010 | 2010 |
| 20100106 | 2010/01/06 | 2    | January | Q1 2010 | 2010 |
| 20100107 | 2010/01/07 | 2    | January | Q1 2010 | 2010 |
| 20100108 | 2010/01/08 | 2    | January | Q1 2010 | 2010 |
| 20100109 | 2010/01/09 | 2    | January | Q1 2010 | 2010 |
| 20100110 | 2010/01/10 | 3    | January | Q1 2010 | 2010 |

**Table 3.18 Mr DVD sample data (Dim\_Actor)**

| Actor_SK | ActorCode | ActorName          | DateOfBirth |
|----------|-----------|--------------------|-------------|
| 8        | DIAC0     | Cameron Diaz       | 1972/08/30  |
| 9        | FORH0     | Harrison Ford      | 1942/07/13  |
| 10       | FRAB0     | Brendan Fraser     | 1968/12/03  |
| 11       | FREM0     | Morgan Freeman     | 1937/06/01  |
| 12       | FREP0     | Paul Freeman       | 1943/01/18  |
| 13       | JOLA0     | Angelina Jolie     | 1975/06/04  |
| 14       | KUTA0     | Ashton Kutcher     | 1978/02/07  |
| 15       | LITJ0     | John Lithgow       | 1945/10/19  |
| 16       | MCAJ0     | James McAvoy       | 1979/04/21  |
| 17       | MURE0     | Eddie Murphy       | 1961/04/03  |
| 18       | MYEM0     | Mike Myers         | 1963/05/25  |
| 19       | PLAO0     | Oliver Platt       | 1960/01/12  |
| 20       | RUSK0     | Keri Russel        | 1976/03/23  |
| 21       | STAS0     | Sylvester Stallone | 1946/07/06  |
| 22       | YOUB0     | Burt Young         | 1940/04/30  |

**Table 3.19 Mr DVD sample data (Dim\_ProdCom)**

| ProdCom_SK | CompanyCode | CompanyName        | City        |
|------------|-------------|--------------------|-------------|
| 26         | 20T         | 20th Century Fox   | Los Angeles |
| 27         | COL         | Columbia Pictures  | Culver City |
| 28         | DRE         | DreamWorks         | Glendale    |
| 29         | MGM         | MGM Studios        | Los Angeles |
| 30         | PAR         | Paramount Pictures | Hollywood   |
| 31         | SON         | Sony Pictures      | Culver City |
| 32         | UNI         | Universal Studios  | Los Angeles |

**Table 3.20 Mr DVD sample data (Fact\_Sales)**

| Date_SK  | Customer_SK | Employee_SK | DVD_SK | ProdCom_SK | OrderID   | Quantity | Revenue |
|----------|-------------|-------------|--------|------------|-----------|----------|---------|
| 20100212 | 47          | 35          | 69     | 27         | MRDVD2036 | 1        | 230.95  |
| 20100212 | 47          | 35          | 71     | 32         | MRDVD2036 | 1        | 170.95  |
| 20100216 | 50          | 36          | 67     | 30         | MRDVD2037 | 2        | 321.50  |
| 20100525 | 52          | 41          | 70     | 28         | MRDVD2038 | 1        | 130.55  |
| 20100525 | 52          | 41          | 72     | 26         | MRDVD2038 | 1        | 210.85  |
| 20100613 | 46          | 40          | 64     | 31         | MRDVD2039 | 1        | 160.95  |
| 20100613 | 46          | 40          | 72     | 26         | MRDVD2039 | 1        | 210.85  |
| 20100825 | 51          | 34          | 69     | 27         | MRDVD2044 | 2        | 461.90  |
| 20100908 | 50          | 41          | 64     | 31         | MRDVD2045 | 1        | 160.95  |
| 20100908 | 50          | 41          | 65     | 26         | MRDVD2045 | 1        | 180.85  |
| 20100908 | 50          | 41          | 68     | 29         | MRDVD2045 | 1        | 160.95  |
| 20100922 | 46          | 34          | 71     | 32         | MRDVD2051 | 1        | 170.95  |
| 20100922 | 46          | 41          | 66     | 31         | MRDVD2052 | 1        | 210.95  |
| 20100922 | 46          | 41          | 71     | 32         | MRDVD2052 | 1        | 170.95  |

Dimensional modelling is centred on two key components which are called facts and attributes. The model is also commonly known as a star schema or a data mart. The name, star schema, is derived from the physical appearance of the model which has a central table, called a fact table, which houses facts, and a set of surrounding tables, called dimension tables, that houses attributes. The shape of a dimensional model is depicted in figure 3.4 below. In figure 3.3 above, Fact\_Sales, is an example of a fact table, while Dim\_Date is an example of a dimension table.



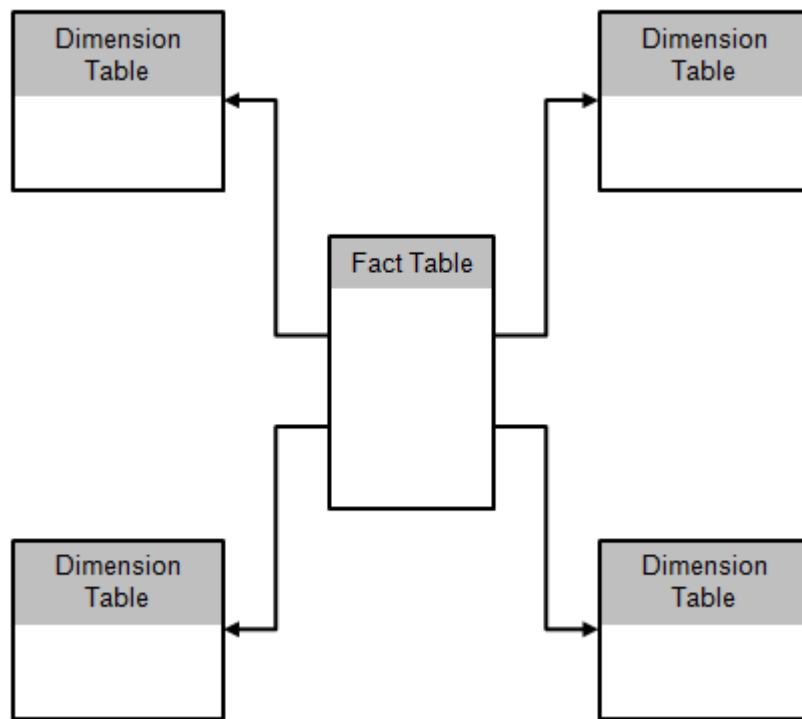


**Figure 3.4 Star schema or dimensional model**

### **3.5.1 Facts and fact tables**

Facts may be defined as the performance measurements of an organisation (Kimball *et al.*, 2008:238). Revenue and quantity are both examples of facts from the Mr DVD example. Facts on their own are of no use to any organisation unless they are provided with some context. Dimensions are the tables in the model that provide this context to the facts. This is best demonstrated by an example. The information that the revenue amounted to R170,95 would not be very meaningful to Joe Lennon unless some context were provided. Joe would want to know the time period to which the revenue relates. Was this revenue generated by all the employees, or a specific employee? Does it relate to a specific DVD category, title, or the entire product range? In the example above, the attributes that would provide context to the revenue fact are month, employee, title, and title category. The information that can now be given to Joe Lennon is that the revenue generated by Jonty Anderson for September 2010 through the sale of an action DVD called Wanted was R170.95.

According to Kimball *et al.* (2008:193), facts belong to one of four distinct categories:

1. additive
2. semi-additive
3. non-additive
4. textual.

Both of the facts in the Mr DVD example, namely, revenue and quantity, are examples of additive facts. Additive facts are always numeric, and may be summed across all attributes. Revenue can be summed across employees, DVD categories and months. On the other hand, it is not possible to add semi-additive facts across all attributes, especially not time-related attributes. Facts such as account balances or inventory levels are semi-additive. These facts are usually snapshot values at a particular point in time. Inventory levels at the end of a specific month may be added across DVD categories in order to obtain the total inventory for the month. However, it is not possible to add inventory levels across months in order to obtain the annual inventory. Non-additive facts are those that cannot be added across any attributes. Temperature or percentages are examples of non-additive facts. Textual facts are the only category of facts that is not numeric. They are extremely although they do exist.

In a dimensional model, facts are grouped together into tables called fact tables. These tables, typically, contain one or more facts relating to a specific business process, for example, sales. As with any relational model, both fact and dimension tables have primary keys. The primary key of a fact table is a composite key consisting of the foreign keys supplied by all the dimension tables that join (or gives context) to a particular fact table. In the Mr DVD example, the primary key of Fact\_Sales is a composite key consisting of Date\_SK, Customer\_SK, Employee\_SK, DVD\_SK, and ProdCom\_SK. The primary keys of dimension tables will be discussed in greater detail later.

### **3.5.2 Attributes and dimension tables**

According to Kimball *et al.* (2008:241), dimension tables consist of three components:

1. attributes
2. hierarchies
3. a primary key.

Attributes assist a dimension in providing context to a fact. The attributes contained in a dimension are usually common to a specific entity in a particular subject area. Examples of entities in subject areas are time and product (or DVD in the Mr DVD example). The time dimension may consist of day, week, month, quarter, year, etc. attributes. Attributes in dimensions frequently have a hierarchy associated with them. In the time dimension mentioned above, the lowest level attribute is that of day. The month attribute is a parent of day, but also a child of quarter. Figure 3.5 below illustrates a hierarchy within a time dimension. These hierarchies make drilling up and down within dimensional models possible. Drilling down involves adding lower level attributes to a report to show the facts at lower levels of detail, while

drilling/rolling up involves the removal of attributes from a report to show the facts at a higher, aggregated level.

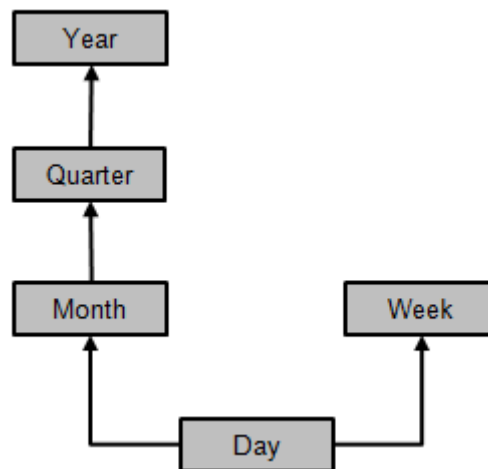


**Figure 3.5 Time dimension hierarchy**

Dimension tables have a special type of primary key, called a surrogate key. Surrogate keys are “meaningless (also known as not-meaningful, artificial, sequence number) keys, typically defined as an integer data type, and sequentially assigned by the data warehouse team to serve as the primary keys of the dimension tables” (Becker, 2006). An example of a primary key is Employee\_SK in Dim\_Employee (see figure 3.3 and table 3.16 above). Primary keys that have a hidden or encoded meaning are called smart or intelligent keys, and should be avoided. As an example, consider a smart key KruV19710221 for a customer record where the customer’s name is Van Zyl Kruger and the date of birth is 21 February 1971. Kimball *et al.* (2008:242) suggest several reasons for insisting that surrogate keys be used:

1. In view of the large volume of data involved in data warehouse queries relative to OLTP, performance considerations are of high importance in data warehouse models. Joins on integer keys perform faster than joins on alphanumeric keys.
2. There are several problems that may arise when production keys are used. The data in the data warehouse is sourced from multiple sources. This, in turn, means that a particular data element may have different keys in different sources. Thus, assigning a single, meaningless integer key, in other words, a surrogate key, to that particular data element resolved this issue. It may also happen that two different data elements have exactly the same production key. Although this key is unique in each of the source systems, it is duplicated when integrated into the warehouse. Another pitfall is that some production systems reuse production keys after some time. In view of the fact that the

data warehouse keeps historical data, an identical key would come into the warehouse at different times for different data elements.

3. Surrogate keys play an essential role in slowly changing dimensions (SCDs). Although the data in dimension tables are relatively stable, it does occasionally change. Slowly changing dimensions constitute a mechanism that keeps track of these changes. Slowly changing dimensions will be discussed in more detail later in this chapter.
4. In view of the fact surrogate keys are 4-byte integers, they take up significantly less storage space than alphanumeric keys.

Kimball *et al.* (2008:254) allow one exception in terms of which smart keys are allowed. Surrogate keys for the date dimension may be in the format of CCYYMMDD. This key is clearly smart, as there is meaning encoded in the key. The motivation for this exception may be attributed to performance considerations. Since one of the components of the fact table's primary key is the foreign key from the date dimension, the smart key may be used to partition a fact table according to dates, for example, one partition per month, per quarter, and so on. The Date\_SK field in Dim\_Date is an example of a smart key in the Mr DVD example is. See table 3.17 for sample data.

Table 3.21 below compares the differences between facts and dimension.

**Table 3.21 Comparison of facts and dimensions**

| Facts                                                                                                                                                                          | Dimensions                                                                                                                                   |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| It is not possible for facts to exist without a business event or transaction occurring. For example, there may not be a revenue amount without a sales transaction occurring. | It is possible for dimensions to exist without a business event occurring. For example, DVDs do exist without a sales transaction occurring. |
| The fact is the verb in a business event.                                                                                                                                      | The dimension is the noun in the business event.                                                                                             |
| Facts are usually numeric values.                                                                                                                                              | Dimensions are usually textual.                                                                                                              |
| Facts are measurements of a business event.                                                                                                                                    | Dimensions give context to the measurements of a business event.                                                                             |

### 3.5.3 Four-step process for dimensional modelling

Kimball *et al.* (2008:246) describe what they call the “four-step dimensional design process” to creating a dimensional model. The steps include

1. choosing the data mart
2. declaring the grain

3. identifying the dimensions
4. identifying the facts.

There are two considerations when choosing the data mart, namely, complexity and priority. Priority is determined by the business while complexity depends partly on whether the data mart is a single-source or multiple-source data mart (Kimball *et al.*, 1998:194). As the name implies, the data for a single-source data mart comes from a single data source whereas the data for a multiple-source data mart comes from more than one source system. Initially it is recommended that single-source data marts be chosen as the complexity and time to implement these single-source data marts is significantly lower than for multiple-source data marts. Kimball *et al.* (2008:249) introduce the enterprise data warehouse bus matrix as another tool that may be used to provide an indication of complexity.

An example of an enterprise data warehouse bus matrix is presented table 3.22 below. The matrix contains subject areas in the rows and dimensions in terms of which each of those subject areas need to be analysed in the columns. Each intersection at which a dimension is applicable to a subject area is marked with an "X". The researcher uses an adapted version of the enterprise data warehouse bus matrix which simply adds row and column totals. The row totals provide a clear indication of the complexity of a subject area (the number of dimensions included) while the column totals are an indication of the importance of particular dimensions (how many times a dimension is used). The Mr DVD example data comes from a single data source system only, called DVD POS. Accordingly, any data mart would work. The example of the four-step dimensional modelling process chose a sales data mart.

**Table 3.22 Enterprise data warehouse bus matrix example**

|                 | Date | DVD | Actor | Customer | Employee | Production Company | Dimension 7 | Dimension 8 | ⋮ | Dimension n | Total |
|-----------------|------|-----|-------|----------|----------|--------------------|-------------|-------------|---|-------------|-------|
| Sales           | X    | X   | X     | X        | X        | X                  |             |             |   |             | 6     |
| Inventory       | X    | X   | X     |          |          | X                  |             |             | X | X           | 6     |
| Human Resources | X    |     |       |          | X        |                    | X           | X           | X |             | 5     |
| Subject Area 4  | X    |     |       | X        |          |                    | X           | X           | X |             | 5     |
| ⋮               | X    | X   | X     |          |          |                    |             | X           | X | X           | 6     |
| Subject Area n  | X    |     |       |          |          |                    | X           |             |   |             | 2     |
| Total           | 6    | 3   | 3     | 2        | 2        | 2                  | 3           | 3           | 4 | 2           |       |

The second step in the dimensional design process involves declaring the grain or granularity of the fact table. Granularity refers to the level of detail at which the data is stored in the fact table and it is usually described in the following manner: *fact per dimension\_1 per dimension\_2 ... per dimension\_n*. The lower the grain of the fact table, the better. It is always possible to aggregate fact tables at a low grain to a higher level. On the other hand, it is not possible to degrade highly summarised fact tables (fact tables with a high granularity) to lower levels. Accordingly, it is not possible to report on facts stored at a monthly grain at a daily level. The grain for the Mr DVD example fact table was sales measures per day, customer, employee, DVD and production company.

The third step in the process involves determining the dimensions. Dimensions are always determined by the grain and are usually named in the grain declaration. Dimensions specify the who, what, where, when and why of the facts. From the grain declared in the above section the dimensions for the Mr DVD example may be identified as time (when), customer (who), employee (who), DVD (what), and production company (who).

The last step in the dimensional design process involves selecting the facts. Most facts may easily be identified by looking for numeric values. If a numeric value is a measurement for each instance at the fact table grain, it is most certainly a fact. The only numeric values related to

sales in the Mr DVD example are quantity and revenue. Thus, both of these are examples of facts.

The next sections will briefly discuss some more advanced dimensional modelling concepts, including conformed dimensions and stove pipes, snowflaking, slowly changing dimensions, many-to-many dimension relationships and, finally, ragged hierarchies.

### 3.5.4 Conformed dimensions and stove pipes

In order to ensure that the individual data marts integrate seamlessly, Kimball *et al.* (2008:245) prescribe the use of conformed dimensions. A conformed dimension is a dimension that is designed in such a way that the same dimension may be used to provide context to multiple fact tables. Figure 3.6 below illustrates the way in which conformed dimensions integrate different data marts. It extends the Mr DVD example by adding another fact table, namely, Fact\_Cost. The granularity of this fact table comprises date and cost item. The two fact tables, Fact\_Sales and Fact\_Cost, have one dimension table in common (Dim\_Time). In view of the fact that the dimension is conformed, it may be used to join the two fact tables. Data marts without conformed dimensions are known as stove pipes. These data marts end up as standalone pockets of information that cannot be integrated and should, thus, be avoided.

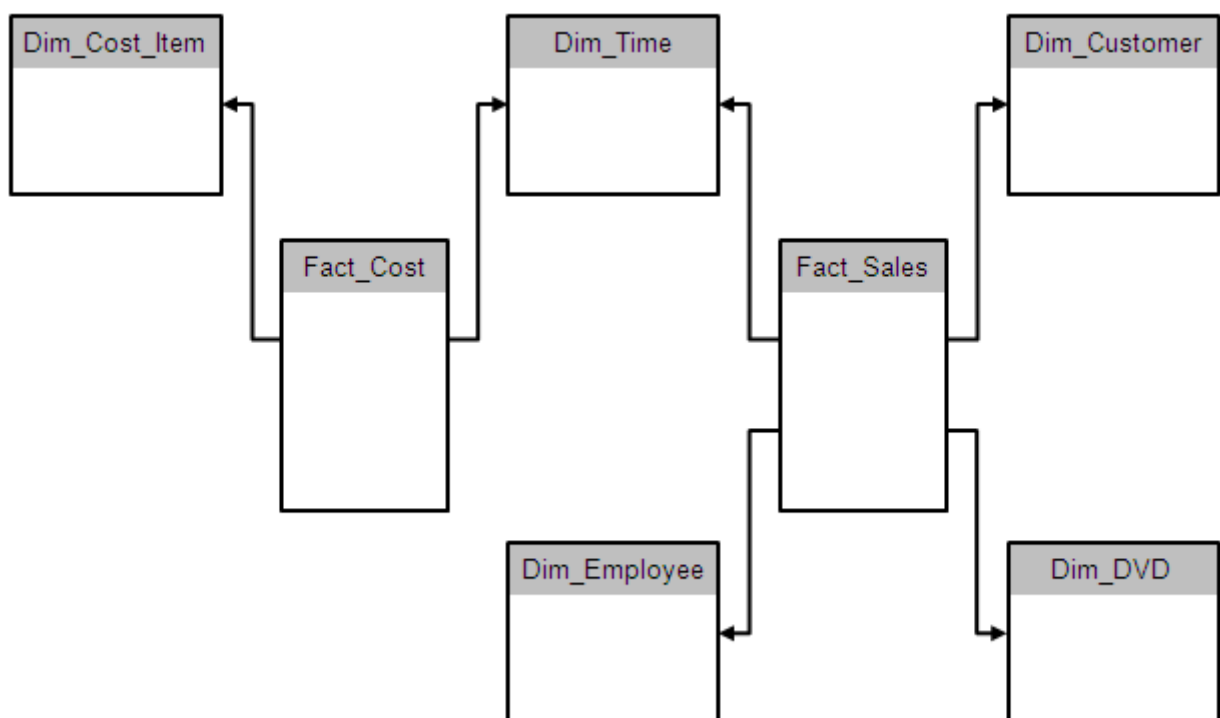


**Figure 3.6 Conformed time and store dimensions**

### 3.5.5 Snowflaking

OLTP system data modellers will feel impelled to normalise the dimension tables. Normalising dimensions is known as snowflaking. Figure 3.7 below presents an example of a single snowflaked dimension. It is termed snowflaking because a fact table with multiple normalised dimensions resembles the shape of a snowflake. Kimball *et al.* (2008:267) list increased complexity and degrading query performance as a result of additional joins as reasons for avoiding snowflaking.

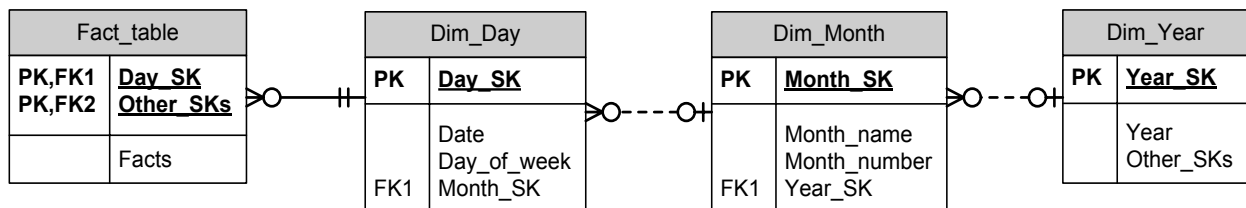


**Figure 3.7 Snowflaked date dimension**

### 3.5.6 Slowly changing dimensions

The next topic to be discussed as regards the subject of dimensional modelling is the concept of slowly changing dimensions. Descriptive attributes in dimension tables frequently change over time. The telephone number of a customer in the Mr DVD example that changes is an example of such a change. There are three options for dealing with these changes; namely, type 1, type 2, and type 3 (Kimball *et al.*, 2008:388). Each of these is best explained by expanding upon the Mr DVD example.

The type 1 option overwrites the existing attribute value with the new value. This method is used for overwriting incorrect values, or when there is no business requirement to retain the history of all the changes. An example from the Mr DVD scenario would be an incorrect address for a customer. When the address is updated in the source system and loaded into the warehouse the previously incorrect value is simply overwritten. A second example is where a customer's telephone number changes. The business may decide that there is no need to keep track of the previous telephone numbers of customers and the old number will simply be overwritten with the new telephone number.

The type 2 option is used when there is a business requirement to track the changes that attribute values undergo over time. As an example, consider an employee whose reporting line changes. Figure 3.8 below presents a simplified dimensional model for the Mr DVD example and table 3.23 the sample source data for this example. Note that from 17 June 2010 Zané



reports to Peter instead of to Jonty. It may be a business requirement to measure the performance of each manager accurately according to the revenue generated by his subordinates over time. The revenue that may be attributed to Jonty over the time period in question amounts to R341.40. However, if a type 1 change is made to the employee dimension, the history will be “changed”. A type 1 change will create the impression that Zané was reporting to Peter all the time, and incorrectly allocate no revenue to Jonty. Type 2 slowly changing dimensions create a new record in the dimension table that contains the altered information. Old records are end dated while a current indicator may also be used.

Table 3.24, table 3.25, and table 3.26 below depict the initial Dim\_Employee dimension table, the Dim\_Employee dimension table after the type 2 change, and the fact table using the type 2 dimension respectively. By joining the fact table to the type 2 dimension it is possible to group correctly by manager and to obtain the correct sales total for Jonty. Using the StartDate and EndDate fields in the dimension also makes it possible to view any information as at a specific point in history.

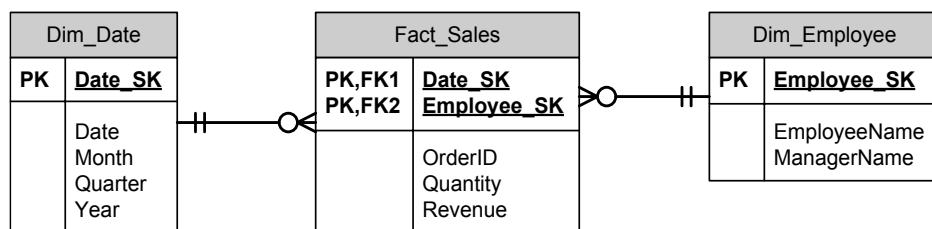


**Figure 3.8 Type 2 slowly changing dimension model**

**Table 3.23 Sample source data: sales amount per month, store and manager**

| Date       | ManagerName | EmployeeName | Revenue |
|------------|-------------|--------------|---------|
| 2010/06/09 | Peter       | Leonard      | 321.50  |
| 2010/06/12 | Jonty       | Zané         | 130.55  |
| 2010/06/13 | Jonty       | Zané         | 210.85  |
| 2010/06/13 | Peter       | Elmarie      | 160.95  |
| 2010/06/14 | Peter       | Elmarie      | 210.85  |
| 2010/06/17 | Peter       | Zané         | 160.95  |
| 2010/06/18 | Peter       | Zané         | 180.85  |
| 2010/06/19 | Peter       | Zané         | 160.95  |
| 2010/06/20 | Peter       | Zané         | 381.90  |

**Table 3.24 Type 2 slowly changing dimension before changes**

| Employee_SK | EmployeeName | ManagerName | StartDate  | EndDate | CurrentIndicator |
|-------------|--------------|-------------|------------|---------|------------------|
| 3           | Leonard      | Peter       | 1900/01/01 | null    | Y                |
| 7           | Elmarie      | Peter       | 1900/01/01 | null    | Y                |
| 8           | Zané         | Jonty       | 1900/01/01 | null    | Y                |

**Table 3.25 Type 2 slowly changing dimension after changes**

| Employee_SK | EmployeeName | ManagerName | StartDate  | EndDate    | CurrentIndicator |
|-------------|--------------|-------------|------------|------------|------------------|
| 3           | Leonard      | Peter       | 1900/01/01 | null       | Y                |
| 7           | Elmarie      | Peter       | 1900/01/01 | null       | Y                |
| 8           | Zané         | Jonty       | 1900/01/01 | 2010/06/16 | N                |
| 9           | Zané         | Peter       | 2010/06/17 | null       | Y                |

**Table 3.26 Fact\_Sales using type 2 slowly changing dimension**

| Date_SK  | Employee_SK | Revenue |
|----------|-------------|---------|
| 20100619 | 3           | 321.50  |
| 20100612 | 8           | 130.55  |
| 20100613 | 8           | 210.85  |
| 20100613 | 7           | 160.95  |
| 20100614 | 7           | 210.85  |
| 20100617 | 9           | 160.95  |
| 20100618 | 9           | 180.85  |
| 20100619 | 9           | 160.95  |
| 20100620 | 9           | 381.90  |

The last slowly changing dimension type is the type 3 option. Type 3 slowly changing dimensions contribute an additional value to the dimension table. An example from the Mr DVD scenario is that of the city field in the Dim\_Customer dimension. The names of several cities in South Africa are being changed. However, people are often not aware of these new names. Accordingly, Joe Lennon may require that it should still be possible to analyse sales information by the old city names as well as the new ones. Type 3 slowly changing dimensions add a new city field to the customer dimension. The old value is moved to this new field, while the new value overwrites the city name. Figure 3.9 below demonstrates this process.

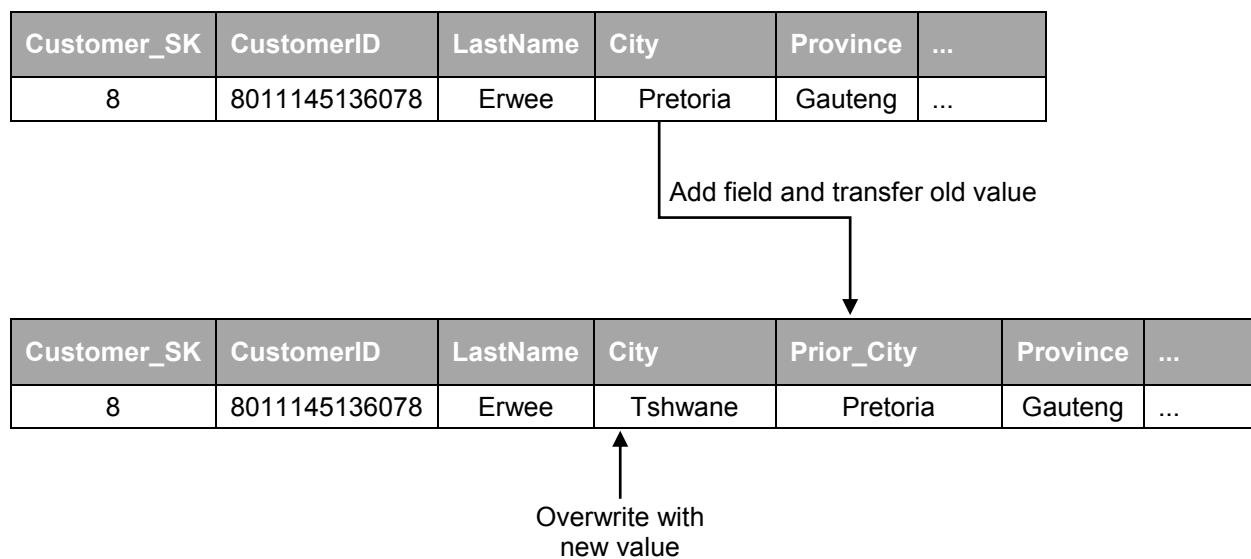


**Figure 3.9 Mr DVD type 3 slowly changing dimension example**

It is important to note that both type 1 and type 2 slowly changing dimensions affect records that changed in a dimension table only, whereas, because of the addition of a new field, type 3 slowly changing dimensions affect every record in a dimension table, whether it changed or not.

### 3.5.7 Many-to-many dimension relationships

Situations are frequently encountered where a many-to-many relationship between a dimension table and the fact table exists instead of the normal one-to-many relationship. In an effort to resolve the problem, Kimball et al. (2008:272) propose a table that they call a bridge table. However, a bridge table is nothing more than an associative entity in relational database terms. Actors provide an example of a many-to-many dimension relationship from the Mr DVD example. Multiple actor records are associated with each line item (the grain of the fact table) because multiple actors play a role in each DVD title. However, this may be resolved by placing a bridge table between the DVD and actor dimension tables, as is illustrated below in figure 3.10.

The primary key of the bridge table is a composite key comprising the surrogate keys of the dimensions involved. The bridge table also contains a weighting factor (Kimball et al., 2008:271). The weighting factor should sum to 1 for all the actors related to a specific DVD. Every additive and semi-additive fact in the fact table should be multiplied with the weighting factor when joining the fact table to the actor dimension table through the bridge table. This, in

turn, will ensure that the facts are not double counted when joining to the bridge table. The allocation of weights is a business issue. In the Mr DVD example, leading actors may have a higher weight than supporting actors.

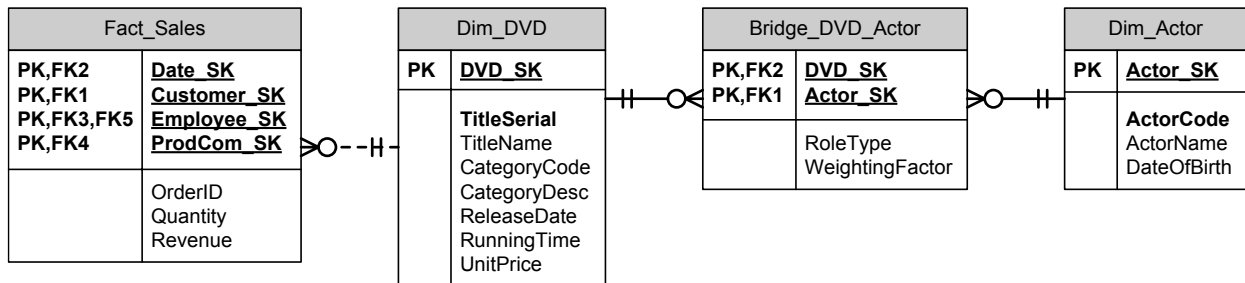


**Figure 3.10 Bridge table example for resolving many-to-many dimension relationships**

### 3.5.8 Ragged hierarchies

Bridge tables are also used to implement ragged hierarchies (Kimball *et al.*, 2008:268). Ragged hierarchies are hierarchies of variable depth. The organisational structure (employees and managers) of MR DVD is an example of a ragged hierarchy from the Mr DVD example. The use of a bridge table in this example is depicted in figure 3.11 below. The employee dimension may be joined directly to the fact table, thus allowing grouping by employees. However, in order to report on the revenue for which each manager is responsible, the employee dimension is joined to the fact table through the bridge table.

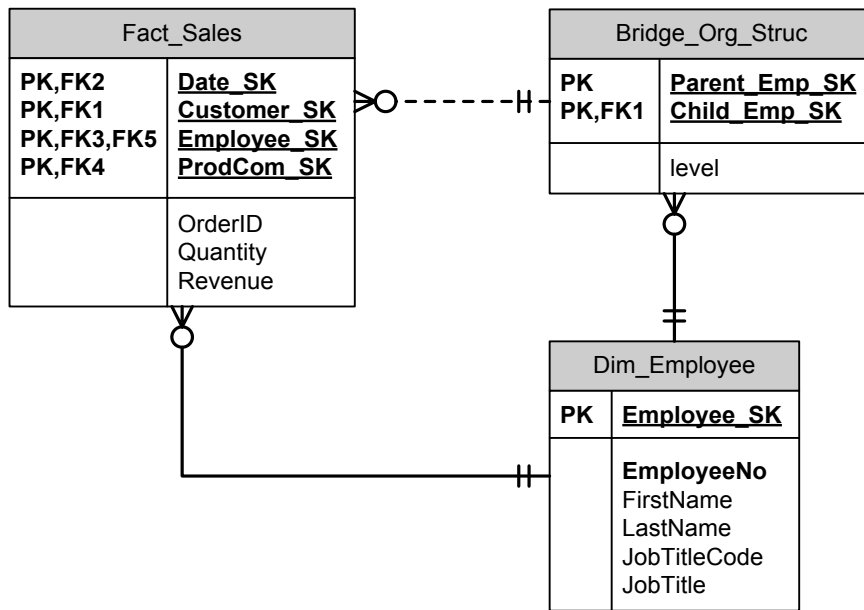


**Figure 3.11 Bridge table example implementing ragged hierarchies**

According to Kimball *et al.* (2008:270,273), the use of bridge tables for both many-to-many dimension relationships as well as ragged hierarchies should be used as a last resort only as it impacts negatively on ease of understanding, usability and performance.

### 3.5.9 Dimensional modelling indexing strategies

In view of the stable design of dimensional models, Kimball *et al.* (2008:347) recommend certain initial indexing strategies. They mention three types of index, namely, B-tree indexes, clustered indexes, and bitmap indexes (Kimball *et al.* 2008:345). Kimball *et al.* (2008:347) recommend an initial indexing strategy which involves clustered indexes on the surrogate key of each dimension as well as bitmap indexes low cardinality fields on the dimension tables that are frequently filtered on. The indexing strategy that is used is also determined by the relational database management system (RDBMS) that is used. SQL Server 2008 R2 is the RDBMS used in this study. SQL Server 2008 R2 does not have bitmap indexes (Microsoft, 2011). However, where bitmap indexes are not available Kimball *et al.* (2008:3487) suggest using B-tree (non-clustered) indexes. For fact tables, Kimball *et al.* (2008:348) recommend a composite clustered index on the composite key of the fact table. They recommend that the date surrogate key be the first key in the index. Additionally, Kimball *et al.* (2008:348) recommend single column indexes on all fact table foreign keys.

This concludes the discussion on dimensional modelling. The following section will focus on the third modelling technique discussed in this paper, namely, data vault modelling.

### 3.6 Data vault modelling

Data vault modelling was invented by Dan E. Linstedt (Linstedt, 2002) and first emerged in 2002. It was designed to address what Linstedt (2002) described as the “performance and other weakness of 3NF and star schema”. Linstedt *et al.* (2009a:26) also noted that both 3NF and the dimensional models lacked both scalability and the flexibility to adapt to business change, as well as the agility of the development team to respond to new requests. The unofficial motto of the data vault is “100% of the data 100% of the time”. This, combined with a strong focus on auditability, means that the data vault has the ability to recreate any of the sources that populate it at any point in time.

The data vault model for Mr DVD is presented in figure 3.12 below. This model will be used to explain the concepts mentioned in the section on data vault modelling. The sample data for the tables is presented in table 3.27 to table 3.51 below.

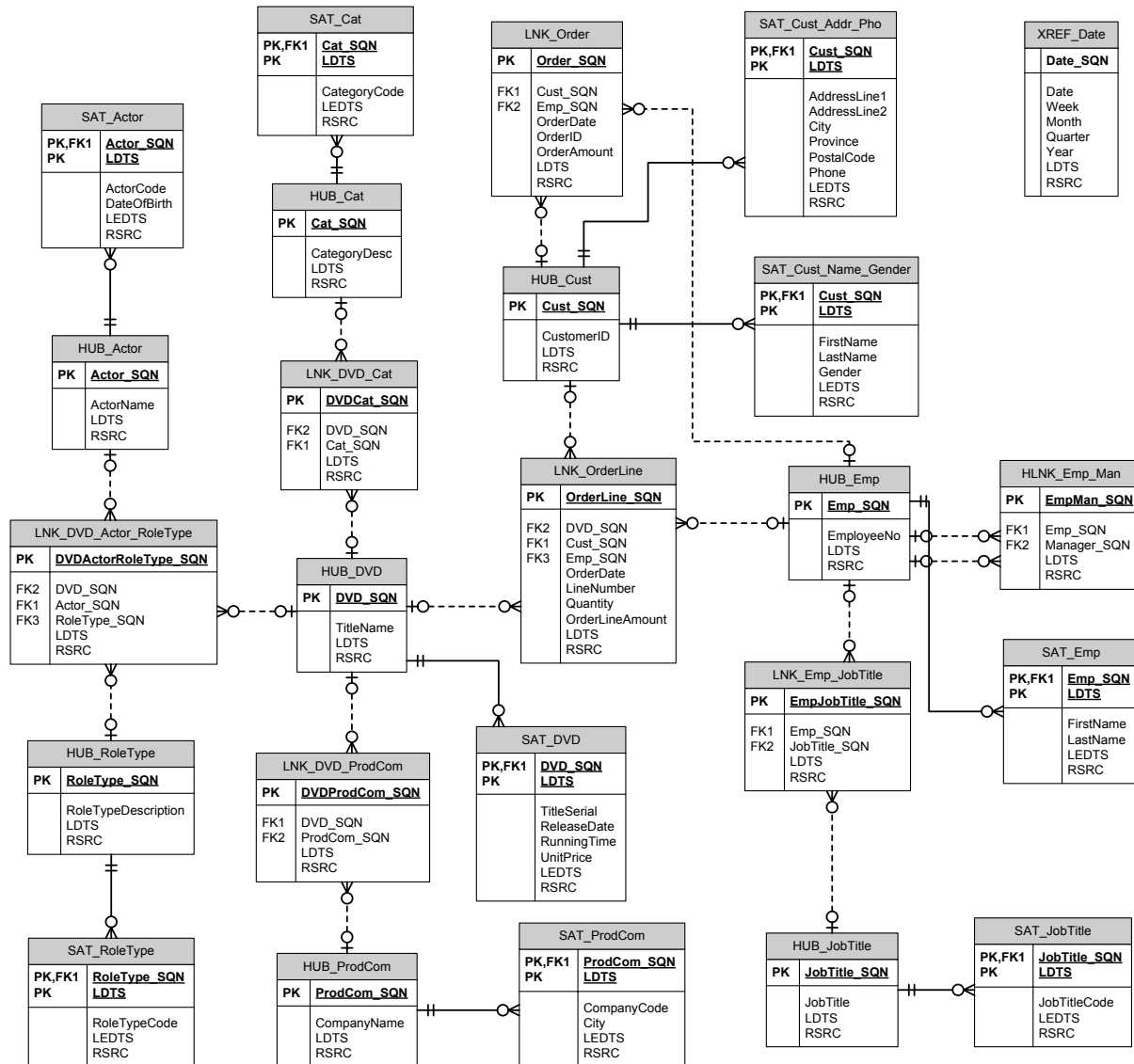


Figure 3.12 Data vault data model for the Mr DVD example

**Table 3.27 Mr DVD sample data (HUB\_Actor)**

| Actor_SQN | ActorName          | LDTS       | RSRC    |
|-----------|--------------------|------------|---------|
| 5         | Karen Allen        | 2010/10/19 | DVD.POS |
| 6         | Andre Braugher     | 2010/10/19 | DVD.POS |
| 7         | Cameron Diaz       | 2010/10/19 | DVD.POS |
| 8         | Harrison Ford      | 2010/10/19 | DVD.POS |
| 9         | Brendan Fraser     | 2010/10/19 | DVD.POS |
| 10        | Morgan Freeman     | 2010/10/19 | DVD.POS |
| 11        | Paul Freeman       | 2010/10/19 | DVD.POS |
| 12        | Angelina Jolie     | 2010/10/19 | DVD.POS |
| 13        | Ashton Kutcher     | 2010/10/19 | DVD.POS |
| 14        | John Lithgow       | 2010/10/19 | DVD.POS |
| 15        | James McAvoy       | 2010/10/19 | DVD.POS |
| 16        | Eddie Murphy       | 2010/10/19 | DVD.POS |
| 17        | Mike Myers         | 2010/10/19 | DVD.POS |
| 18        | Oliver Platt       | 2010/10/19 | DVD.POS |
| 19        | Keri Russel        | 2010/10/19 | DVD.POS |
| 20        | Sylvester Stallone | 2010/10/19 | DVD.POS |
| 21        | Burt Young         | 2010/10/19 | DVD.POS |

**Table 3.28 Mr DVD sample data (SAT\_Actor)**

| Actor_SQN | LDTS       | ActorCode | DateOfBirth | LEDTS | RSRC    |
|-----------|------------|-----------|-------------|-------|---------|
| 5         | 2010/10/19 | ALLK0     | 19511005    | NULL  | DVD.POS |
| 6         | 2010/10/19 | BRAA0     | 19620701    | NULL  | DVD.POS |
| 7         | 2010/10/19 | DIAC0     | 19720830    | NULL  | DVD.POS |
| 8         | 2010/10/19 | FORH0     | 19420713    | NULL  | DVD.POS |
| 9         | 2010/10/19 | FRAB0     | 19681203    | NULL  | DVD.POS |
| 10        | 2010/10/19 | FREM0     | 19370601    | NULL  | DVD.POS |
| 11        | 2010/10/19 | FREP0     | 19430118    | NULL  | DVD.POS |
| 12        | 2010/10/19 | JOLA0     | 19750604    | NULL  | DVD.POS |
| 13        | 2010/10/19 | KUTA0     | 19780207    | NULL  | DVD.POS |
| 14        | 2010/10/19 | LITJ0     | 19451019    | NULL  | DVD.POS |
| 15        | 2010/10/19 | MCAJ0     | 19790421    | NULL  | DVD.POS |
| 16        | 2010/10/19 | MURE0     | 19610403    | NULL  | DVD.POS |
| 17        | 2010/10/19 | MYEM0     | 19630525    | NULL  | DVD.POS |
| 18        | 2010/10/19 | PLAO0     | 19600112    | NULL  | DVD.POS |
| 19        | 2010/10/19 | RUSK0     | 19760323    | NULL  | DVD.POS |
| 20        | 2010/10/19 | STAS0     | 19460706    | NULL  | DVD.POS |
| 21        | 2010/10/19 | YOUB0     | 19400430    | NULL  | DVD.POS |



**Table 3.29 Mr DVD sample data (HUB\_Cat)**

| Cat_SQN | CategoryDesc | LDTS       | RSRC    |
|---------|--------------|------------|---------|
| 15      | Action       | 2010/10/19 | DVD.POS |
| 16      | Adventure    | 2010/10/19 | DVD.POS |
| 17      | Animation    | 2010/10/19 | DVD.POS |
| 18      | Comedy       | 2010/10/19 | DVD.POS |
| 19      | Drama        | 2010/10/19 | DVD.POS |
| 20      | Thriller     | 2010/10/19 | DVD.POS |

**Table 3.30 Mr DVD sample data (SAT\_Cat)**

| Cat_SQN | LDTS       | CategoryCode | LEDTS | RSRC    |
|---------|------------|--------------|-------|---------|
| 15      | 2010/10/19 | ACT          | NULL  | DVD.POS |
| 16      | 2010/10/19 | ADV          | NULL  | DVD.POS |
| 17      | 2010/10/19 | ANI          | NULL  | DVD.POS |
| 18      | 2010/10/19 | COM          | NULL  | DVD.POS |
| 19      | 2010/10/19 | DRA          | NULL  | DVD.POS |
| 20      | 2010/10/19 | THR          | NULL  | DVD.POS |

**Table 3.31 Mr DVD sample data (HUB\_DVD)**

| DVD_SQN | TitleName                      | LDTS       | RSRC    |
|---------|--------------------------------|------------|---------|
| 62      | Cliffhanger                    | 2010/10/19 | DVD.POS |
| 63      | Dr Dolittle                    | 2010/10/19 | DVD.POS |
| 64      | Extra Ordinary Measures        | 2010/10/19 | DVD.POS |
| 65      | Indiana Jones and the Lost Ark | 2010/10/19 | DVD.POS |
| 66      | Rocky                          | 2010/10/19 | DVD.POS |
| 67      | Salt                           | 2010/10/19 | DVD.POS |
| 68      | Shrek                          | 2010/10/19 | DVD.POS |
| 69      | Wanted                         | 2010/10/19 | DVD.POS |
| 70      | What Happens in Vegas          | 2010/10/19 | DVD.POS |

**Table 3.32 Mr DVD sample data (SAT\_DVD)**

| DVD_SQN | LDTs       | TitleSerial | RunningTime | ReleaseDate | UnitPrice | LEDTS | RSRC    |
|---------|------------|-------------|-------------|-------------|-----------|-------|---------|
| 62      | 2010/10/19 | CLIFFHAN00  | 113         | 19971126    | 160.95    | NULL  | DVD.POS |
| 63      | 2010/10/19 | DRDOLITT00  | 85          | 19990803    | 180.85    | NULL  | DVD.POS |
| 64      | 2010/10/19 | EXTRAORD00  | 106         | 20100122    | 210.95    | NULL  | DVD.POS |
| 65      | 2010/10/19 | INDIANAJ00  | 115         | 20080513    | 160.75    | NULL  | DVD.POS |
| 66      | 2010/10/19 | ROCKY00000  | 119         | 20050208    | 160.95    | NULL  | DVD.POS |
| 67      | 2010/10/19 | SALT000000  | 103         | 20100723    | 230.95    | NULL  | DVD.POS |
| 68      | 2010/10/19 | SHREK00000  | 90          | 20100426    | 130.55    | NULL  | DVD.POS |
| 69      | 2010/10/19 | WANTED0000  | 110         | 20080627    | 170.95    | NULL  | DVD.POS |
| 70      | 2010/10/19 | WHATHAPP00  | 99          | 20080826    | 210.85    | NULL  | DVD.POS |

**Table 3.33 Mr DVD sample data (HUB\_Emp)**

| Emp_SQN | EmployeeNo | LDTs       | RSRC    |
|---------|------------|------------|---------|
| 128     | AndeJ04    | 2010/10/19 | DVD.POS |
| 129     | AlisP02    | 2010/10/19 | DVD.POS |
| 130     | OgilL08    | 2010/10/19 | DVD.POS |
| 131     | LennJ01    | 2010/10/19 | DVD.POS |
| 132     | JohnK15    | 2010/10/19 | DVD.POS |
| 133     | RobbS14    | 2010/10/19 | DVD.POS |
| 134     | VanSE09    | 2010/10/19 | DVD.POS |
| 135     | VanSZ12    | 2010/10/19 | DVD.POS |

**Table 3.34 Mr DVD sample data (SAT\_Emp)**

| Emp_SQN | LDTs       | FirstName | LastName      | LEDTS | RSRC    |
|---------|------------|-----------|---------------|-------|---------|
| 128     | 2010/10/19 | Jonty     | Anderson      | NULL  | DVD.POS |
| 129     | 2010/10/19 | Peter     | Alistair      | NULL  | DVD.POS |
| 130     | 2010/10/19 | Leonard   | Ogilvy        | NULL  | DVD.POS |
| 131     | 2010/10/19 | Joe       | Lennon        | NULL  | DVD.POS |
| 132     | 2010/10/19 | Kevin     | Johnson       | NULL  | DVD.POS |
| 133     | 2010/10/19 | Stephen   | Robbie        | NULL  | DVD.POS |
| 134     | 2010/10/19 | Elmarie   | Van Schalkwyk | NULL  | DVD.POS |
| 135     | 2010/10/19 | Zané      | Van Schalkwyk | NULL  | DVD.POS |

**Table 3.35 Mr DVD sample data (HUB\_JobTitle)**

| JobTitle_SQN | JobTitle         | LDTs       | RSRC    |
|--------------|------------------|------------|---------|
| 87           | Junior Assistant | 2010/10/19 | DVD.POS |
| 88           | Senior Assistant | 2010/10/19 | DVD.POS |
| 89           | Store Manager    | 2010/10/19 | DVD.POS |

**Table 3.36 Mr DVD sample data (SAT\_JobTitle)**

| JobTitle_SQN | LDTs       | JobTitleCode | LEDTS | RSRC    |
|--------------|------------|--------------|-------|---------|
| 87           | 2010/10/19 | J            | NULL  | DVD.POS |
| 88           | 2010/10/19 | S            | NULL  | DVD.POS |
| 89           | 2010/10/19 | M            | NULL  | DVD.POS |

**Table 3.37 Mr DVD sample data (HUB\_ProdCom)**

| ProdCom_SQN | CompanyName        | LDTs       | RSRC    |
|-------------|--------------------|------------|---------|
| 54          | 20th Century Fox   | 2010/10/19 | DVD.POS |
| 55          | Columbia Pictures  | 2010/10/19 | DVD.POS |
| 56          | DreamWorks         | 2010/10/19 | DVD.POS |
| 57          | MGM Studios        | 2010/10/19 | DVD.POS |
| 58          | Paramount Pictures | 2010/10/19 | DVD.POS |
| 59          | Sony Pictures      | 2010/10/19 | DVD.POS |
| 60          | Universal Studios  | 2010/10/19 | DVD.POS |

**Table 3.38 Mr DVD sample data (SAT\_ProdCom)**

| ProdCom_SQN | LDTs       | CompanyCode | City        | LEDTS | RSRC    |
|-------------|------------|-------------|-------------|-------|---------|
| 54          | 2010/10/19 | 20T         | Los Angeles | NULL  | DVD.POS |
| 55          | 2010/10/19 | COL         | Culver City | NULL  | DVD.POS |
| 56          | 2010/10/19 | DRE         | Glendale    | NULL  | DVD.POS |
| 57          | 2010/10/19 | MGM         | Los Angeles | NULL  | DVD.POS |
| 58          | 2010/10/19 | PAR         | Hollywood   | NULL  | DVD.POS |
| 59          | 2010/10/19 | SON         | Culver City | NULL  | DVD.POS |
| 60          | 2010/10/19 | UNI         | Los Angeles | NULL  | DVD.POS |

**Table 3.39 Mr DVD sample data (HUB\_RoleType)**

| RoleType_SQN | RoleTypeDescription | LDTs       | RSRC    |
|--------------|---------------------|------------|---------|
| 45           | Leading             | 2010/10/19 | DVD.POS |
| 46           | Supporting          | 2010/10/19 | DVD.POS |
| 47           | Voice               | 2010/10/19 | DVD.POS |

**Table 3.40 Mr DVD sample data (SAT\_RoleType)**

| RoleType_SQN | LDTs       | RoleTypeCode | LEDTS | RSRC    |
|--------------|------------|--------------|-------|---------|
| 45           | 2010/10/19 | L            | NULL  | DVD.POS |
| 46           | 2010/10/19 | S            | NULL  | DVD.POS |
| 47           | 2010/10/19 | V            | NULL  | DVD.POS |

**Table 3.41 Mr DVD sample data (HUB\_Cust)**

| Cust_SQN | CustomerID    | LDTs       | RSRC    |
|----------|---------------|------------|---------|
| 45       | 6908086310075 | 2010/10/19 | DVD.POS |
| 46       | 7102215895085 | 2010/10/19 | DVD.POS |
| 47       | 7406286136008 | 2010/10/19 | DVD.POS |
| 48       | 7809184852099 | 2010/10/19 | DVD.POS |
| 49       | 7905165036035 | 2010/10/19 | DVD.POS |
| 50       | 7905255005089 | 2010/10/19 | DVD.POS |
| 51       | 8003203652098 | 2010/10/19 | DVD.POS |
| 52       | 8011145136078 | 2010/10/19 | DVD.POS |

**Table 3.42 Mr DVD sample data (SAT\_Cust\_Addr\_Pho)**

| Cust_SQN | LDTs       | AddressLine1         | AddressLine2    | City        | Province | PostalCode | Phone     | LEDTS | RSRC    |
|----------|------------|----------------------|-----------------|-------------|----------|------------|-----------|-------|---------|
| 45       | 2010/10/19 | 436 Aureole Avenue   | NULL            | Midrand     | Gauteng  | 2635       | 843650136 | NULL  | DVD.POS |
| 46       | 2010/10/19 | 23 Duncan Street     | NULL            | Midrand     | Gauteng  | 2635       | 836350698 | NULL  | DVD.POS |
| 47       | 2010/10/19 | 17 Almeisan          | Rough Street    | Randburg    | Gauteng  | 2045       | 797213985 | NULL  | DVD.POS |
| 48       | 2010/10/19 | 139 Sibaya Sands     | Straight Street | Roodepoort  | Gauteng  | 1724       | 723600201 | NULL  | DVD.POS |
| 49       | 2010/10/19 | 188 Ruben Street     | NULL            | Alberton    | Gauteng  | 1983       | 846564138 | NULL  | DVD.POS |
| 50       | 2010/10/19 | 12 Montego Blue      | Kelly Avenue    | Krugersdorp | Gauteng  | 1985       | 836982401 | NULL  | DVD.POS |
| 51       | 2010/10/19 | 66 Victorian Heights | 2 Anvil Street  | Fourways    | Gauteng  | 1885       | 825689340 | NULL  | DVD.POS |
| 52       | 2010/10/19 | 75a Tulbach Road     | NULL            | Pretoria    | Gauteng  | 1368       | 832992791 | NULL  | DVD.POS |

**Table 3.43 Mr DVD sample data (SAT\_Cust\_Name\_Gender)**

| Cust_SQN | LDTs       | FirstName | LastName    | Gender | LEDTS | RSRC    |
|----------|------------|-----------|-------------|--------|-------|---------|
| 45       | 2010/10/19 | Johann    | Kruger      | M      | NULL  | DVD.POS |
| 46       | 2010/10/19 | Van Zyl   | Kruger      | M      | NULL  | DVD.POS |
| 47       | 2010/10/19 | James     | Green       | M      | NULL  | DVD.POS |
| 48       | 2010/10/19 | Elmien    | Geel        | F      | NULL  | DVD.POS |
| 49       | 2010/10/19 | Kevin     | Venter      | M      | NULL  | DVD.POS |
| 50       | 2010/10/19 | Spike     | Moony       | M      | NULL  | DVD.POS |
| 51       | 2010/10/19 | Yolandi   | Langenhoven | F      | NULL  | DVD.POS |
| 52       | 2010/10/19 | Pieter    | Erwee       | M      | NULL  | DVD.POS |

**Table 3.44 Mr DVD sample data (XREF\_Date)**

| Date_SQN | Date       | Week | Month   | Quarter | Year | LDTs       | RSRC   |
|----------|------------|------|---------|---------|------|------------|--------|
| 20100101 | 2010/01/01 | 1    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |
| 20100102 | 2010/01/01 | 1    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |
| 20100103 | 2010/01/01 | 2    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |
| 20100104 | 2010/01/01 | 2    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |
| 20100105 | 2010/01/01 | 2    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |
| 20100106 | 2010/01/01 | 2    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |
| 20100107 | 2010/01/01 | 2    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |
| 20100108 | 2010/01/01 | 2    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |
| 20100109 | 2010/01/01 | 2    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |
| 20100110 | 2010/01/01 | 3    | January | Q1 2010 | 2010 | 2010/10/20 | SYSTEM |

**Table 3.45 Mr DVD sample data (HLNK\_Emp\_Man)**

| EmpMan_SQN | Emp_SQN    | Manager_SQN | LDTs       | RSRC    |
|------------|------------|-------------|------------|---------|
| 6          | 1900/05/07 | 131         | 2010/10/19 | DVD.POS |
| 7          | 1900/05/08 | 131         | 2010/10/19 | DVD.POS |
| 8          | 1900/05/09 | 129         | 2010/10/19 | DVD.POS |
| 9          | 1900/05/11 | 128         | 2010/10/19 | DVD.POS |
| 10         | 1900/05/12 | 128         | 2010/10/19 | DVD.POS |
| 11         | 1900/05/13 | 129         | 2010/10/19 | DVD.POS |
| 12         | 1900/05/14 | 128         | 2010/10/19 | DVD.POS |

**Table 3.46 Mr DVD sample data (LNK\_DVD\_Actor\_RoleType)**

| DVDActorRoleType_SQN | DVD_SQN | Actor_SQN | RoleType_SQN | LDTs       | RSRC    |
|----------------------|---------|-----------|--------------|------------|---------|
| 454                  | 65      | 5         | 46           | 2010/10/19 | DVD.POS |
| 455                  | 67      | 6         | 46           | 2010/10/19 | DVD.POS |
| 456                  | 68      | 7         | 47           | 2010/10/19 | DVD.POS |
| 457                  | 70      | 7         | 45           | 2010/10/19 | DVD.POS |
| 458                  | 64      | 8         | 45           | 2010/10/19 | DVD.POS |
| 459                  | 65      | 8         | 45           | 2010/10/19 | DVD.POS |
| 460                  | 64      | 9         | 45           | 2010/10/19 | DVD.POS |
| 461                  | 69      | 10        | 45           | 2010/10/19 | DVD.POS |
| 462                  | 65      | 11        | 46           | 2010/10/19 | DVD.POS |
| 463                  | 67      | 12        | 45           | 2010/10/19 | DVD.POS |
| 464                  | 69      | 12        | 45           | 2010/10/19 | DVD.POS |
| 465                  | 70      | 13        | 45           | 2010/10/19 | DVD.POS |
| 466                  | 62      | 14        | 46           | 2010/10/19 | DVD.POS |
| 467                  | 68      | 14        | 47           | 2010/10/19 | DVD.POS |
| 468                  | 69      | 15        | 46           | 2010/10/19 | DVD.POS |
| 469                  | 63      | 16        | 45           | 2010/10/19 | DVD.POS |
| 470                  | 68      | 16        | 47           | 2010/10/19 | DVD.POS |
| 471                  | 68      | 17        | 47           | 2010/10/19 | DVD.POS |
| 472                  | 63      | 18        | 46           | 2010/10/19 | DVD.POS |
| 473                  | 64      | 19        | 45           | 2010/10/19 | DVD.POS |
| 474                  | 62      | 20        | 45           | 2010/10/19 | DVD.POS |
| 475                  | 66      | 20        | 45           | 2010/10/19 | DVD.POS |
| 476                  | 66      | 21        | 46           | 2010/10/19 | DVD.POS |

**Table 3.47 Mr DVD sample data (LNK\_DVD\_Cat)**

| DVDCat_SQN | DVD_SQN | Cat_SQN | LDTs       | RSRC    |
|------------|---------|---------|------------|---------|
| 4          | 62      | 15      | 2010/10/19 | DVD.POS |
| 5          | 63      | 18      | 2010/10/19 | DVD.POS |
| 6          | 64      | 19      | 2010/10/19 | DVD.POS |
| 7          | 65      | 16      | 2010/10/19 | DVD.POS |
| 8          | 66      | 15      | 2010/10/19 | DVD.POS |
| 9          | 67      | 20      | 2010/10/19 | DVD.POS |
| 10         | 68      | 17      | 2010/10/19 | DVD.POS |
| 11         | 69      | 15      | 2010/10/19 | DVD.POS |
| 12         | 70      | 18      | 2010/10/19 | DVD.POS |

**Table 3.48 Mr DVD sample data (LNK\_DVD\_ProdCom)**

| DVDProdCom_SQN | DVD_SQN | ProdCom_SQN | LDTs       | RSRC    |
|----------------|---------|-------------|------------|---------|
| 31             | 62      | 59          | 2010/10/19 | DVD.POS |
| 32             | 63      | 54          | 2010/10/19 | DVD.POS |
| 33             | 64      | 59          | 2010/10/19 | DVD.POS |
| 34             | 65      | 58          | 2010/10/19 | DVD.POS |
| 35             | 66      | 57          | 2010/10/19 | DVD.POS |
| 36             | 67      | 55          | 2010/10/19 | DVD.POS |
| 37             | 68      | 56          | 2010/10/19 | DVD.POS |
| 38             | 69      | 60          | 2010/10/19 | DVD.POS |
| 39             | 70      | 54          | 2010/10/19 | DVD.POS |

**Table 3.49 Mr DVD sample data (LNK\_Emp\_JobTitle)**

| EmpJobTitle_SQN | Emp_SQN | JobTitle_SQN | LDTs       | RSRC    |
|-----------------|---------|--------------|------------|---------|
| 17              | 128     | 88           | 2010/10/19 | DVD.POS |
| 18              | 129     | 88           | 2010/10/19 | DVD.POS |
| 19              | 130     | 87           | 2010/10/19 | DVD.POS |
| 20              | 131     | 89           | 2010/10/19 | DVD.POS |
| 21              | 132     | 87           | 2010/10/19 | DVD.POS |
| 22              | 133     | 87           | 2010/10/19 | DVD.POS |
| 23              | 134     | 87           | 2010/10/19 | DVD.POS |
| 24              | 135     | 87           | 2010/10/19 | DVD.POS |



**Table 3.50 Mr DVD sample data (LNK\_Order)**

| Order_SQN | Cust_SQN | Emp_SQN | OrderDate | OrderID   | OrderAmount | LDTs       | RSRC    |
|-----------|----------|---------|-----------|-----------|-------------|------------|---------|
| 345       | 46       | 129     | 20100212  | MRDVD2036 | 401.90      | 2010/10/20 | DVD.POS |
| 346       | 49       | 130     | 20100216  | MRDVD2037 | 321.50      | 2010/10/20 | DVD.POS |
| 347       | 51       | 135     | 20100525  | MRDVD2038 | 341.40      | 2010/10/20 | DVD.POS |
| 348       | 45       | 134     | 20100613  | MRDVD2039 | 371.80      | 2010/10/20 | DVD.POS |
| 349       | 50       | 128     | 20100825  | MRDVD2044 | 461.90      | 2010/10/20 | DVD.POS |
| 350       | 49       | 135     | 20100908  | MRDVD2045 | 502.75      | 2010/10/20 | DVD.POS |
| 351       | 45       | 128     | 20100922  | MRDVD2051 | 170.95      | 2010/10/20 | DVD.POS |
| 352       | 45       | 135     | 20100922  | MRDVD2052 | 381.90      | 2010/10/20 | DVD.POS |

**Table 3.51 Mr DVD sample data (LNK\_OrderLine)**

| OrderLine_SQN | DVD_SQN | Cust_SQN | Emp_SQN | OrderDate | LineNumber | Quantity | OrderLineAmount | LDTs       | RSRC    |
|---------------|---------|----------|---------|-----------|------------|----------|-----------------|------------|---------|
| 9846          | 67      | 46       | 129     | 20100212  | 1          | 1        | 230.95          | 2010/10/20 | DVD.POS |
| 9847          | 69      | 46       | 129     | 20100212  | 2          | 1        | 170.95          | 2010/10/20 | DVD.POS |
| 9848          | 65      | 49       | 130     | 20100216  | 1          | 2        | 321.5           | 2010/10/20 | DVD.POS |
| 9849          | 70      | 51       | 135     | 20100525  | 1          | 1        | 210.85          | 2010/10/20 | DVD.POS |
| 9850          | 68      | 51       | 135     | 20100525  | 2          | 1        | 130.55          | 2010/10/20 | DVD.POS |
| 9851          | 62      | 45       | 134     | 20100613  | 1          | 1        | 160.95          | 2010/10/20 | DVD.POS |
| 9852          | 70      | 45       | 134     | 20100613  | 2          | 1        | 210.85          | 2010/10/20 | DVD.POS |
| 9853          | 67      | 50       | 128     | 20100825  | 1          | 2        | 461.9           | 2010/10/20 | DVD.POS |
| 9854          | 62      | 49       | 135     | 20100908  | 1          | 1        | 160.95          | 2010/10/20 | DVD.POS |
| 9855          | 63      | 49       | 135     | 20100908  | 2          | 1        | 180.85          | 2010/10/20 | DVD.POS |
| 9856          | 66      | 49       | 135     | 20100908  | 3          | 1        | 160.95          | 2010/10/20 | DVD.POS |
| 9857          | 69      | 45       | 128     | 20100922  | 1          | 1        | 170.95          | 2010/10/20 | DVD.POS |
| 9858          | 64      | 45       | 135     | 20100922  | 1          | 1        | 210.95          | 2010/10/20 | DVD.POS |
| 9859          | 69      | 45       | 135     | 20100922  | 2          | 1        | 170.95          | 2010/10/20 | DVD.POS |

The data vault consists of three core components; namely, hub entities (hereafter referred to as hubs), link entities (hereafter referred to as links) and satellite entities (hereafter referred to as satellites). Hubs contain business keys and links contain associations between hubs, while satellites contain descriptive information that provides context to both hubs and links (Linstedt & Graziano, 2010:16). Each of these three components also contains certain common attributes such as sequence numbers, record sources and load dates. The common components are generated during the data vault loading process and are not auditable.

Sequence numbers (SQN) are numeric, meaningless, incremental integer values assigned to individual records by the data vault implementation team. Sequence numbers serve as primary keys for hubs and links and as part of the composite primary key for satellites. The main purpose of sequence numbers is to improve the performance of join operations on the DBMS.

The record source (RSRC) is an alphanumeric field that contains the name of the source system of the data. It plays a key role in the auditability and traceability of the data in the data vault (Linstedt & Graziano, 2010:40). Record sources provide a component to the data vault against which business rules can be validated (Linstedt & Graziano, 2010:43). Consider the Mr DVD example. As stated previously, Mr DVD uses a system known as DVD POS to capture its sales transactions. However, the human resources (HR) department may also use another application, such as Pastel Payroll in order to manage its employee payroll. The business may require that “new employees be created in the Pastel Payroll system only”. However, this rule can be validated by looking at the record source field for all employee records. If the value of the record source field is always “PASTEL.PAYROLL”, then the rule is valid. However, if some employee records have a record source value of “DVD.POS”, then this rule is no longer valid. The record source should always be recorded at the lowest level of granularity; for example “PASTEL.PAYROLL”, indicating that the source row originated from the Pastel accounting system payroll module.

Load dates (LDTS) are another common attribute to each of the data vault components. Load dates are system-generated timestamps that indicate when the data first arrived in the data vault. They can be used to calculate the latency of the arriving data (when the load date is compared to an extraction date) or to reload a particular time frame of data should any problems occur with the loading process (Linstedt & Graziano, 2010:31). Load dates are generated differently depending on whether the data vault is loaded in batch or real-time (Linstedt & Graziano, 2010:32). This difference may be attributed to the fact that load

timestamps for batch data are discrete while load timestamps for real-time data are continuous.

Figure 3.13 below compares batch and real-time timestamps in the data vault. With batch processing, there is one timestamp for every record in the batch and the timestamps between batches are unique. Batch timestamps for each batch can be adjusted to be anytime during the batch load. For example, if the batch load window is usually from 19:00 in the evening to 05:00 the next morning, the load date can be set to any time between 19:00 and 05:00. This time should, however, be consistently adjusted across all batches. With real-time processing, the load date for each record in the data vault is exactly the time that the record arrives in the data vault. No adjustments are made to real-time timestamps as the data does not arrive at fixed time intervals as is the case with batch processing (Linstedt & Graziano, 2010:31).

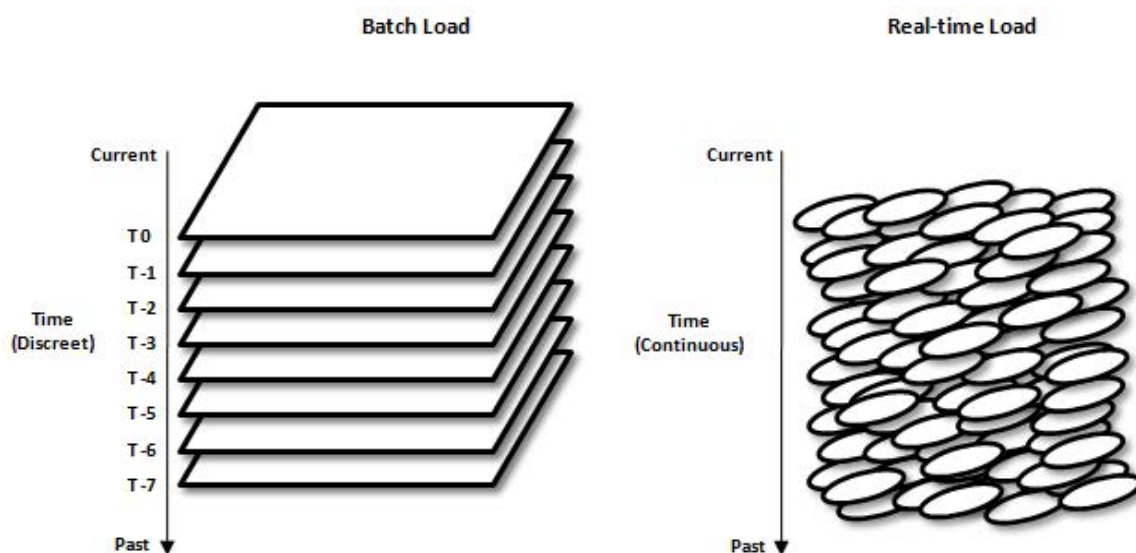


**Figure 3.13 Comparison of batch and real-time timestamps (Linstedt & Graziano, 2010:31)**

Record sources and load dates, two of the mandatory common attributes of hubs, links, and satellites, together with the rule that the data vault should contain 100% of the data from the sources within scope 100% of the time, gives the data vault the capability to generate an exact copy of any of its sources at any point in time.

Linstedt and Graziano (2010:40) also discuss a few other common attributes that can be utilised in hubs, links, and satellites and include sub sequence numbers, load end dates

(LEDTS), extract dates, last seen dates, record creation dates and process IDs. However, these attributes are not mandatory and are not discussed in any detail in this chapter.

The next sections discuss hub, satellite and link entities. This is followed by discussions on link-to-link relationships, bridge tables and cross-reference tables. After describing the seven step process to data vault model design, the section concludes with an initial indexing strategy for the data vault.

### 3.6.1 Hub entities

Hubs are entities that, together with the mandatory common attributes, contain a list of unique business keys. According to Linstedt and Graziano (2010:42), the purpose of a hub is “to track the first time the data vault sees the key arrive in the warehousing load, and where it came from. The Hub is a business key recording device”. Hub entities must always possess the following attributes – as illustrated by figure 3.14 below:

- business key
- sequence number
- load date
- record source.

| HUB |                                            |
|-----|--------------------------------------------|
| PK  | <u>Sequence Number</u>                     |
|     | Business_Key<br>Load_Date<br>Record_Source |

| HUB_Emp |                            |
|---------|----------------------------|
| PK      | <u>Emp SQN</u>             |
|         | EmployeeNo<br>LDTS<br>RSRC |

**Figure 3.14 Hub entity mandatory components and HUB\_Emp example**

A business key refers to data that the business uses to identify information in a unique way. Examples of business keys include the following:

- Identity numbers uniquely identify customers.
- Account numbers uniquely identify account.
- Stock keeping units (SKUs) uniquely identify products.

- Vehicle identification numbers (VIN) uniquely identify vehicles. Note that VIN numbers are different to vehicle registration numbers.

As should be apparent from the list, business keys can be either numeric or alphanumeric, while they always have meaning to the business. These business keys are frequently sequence IDs (primary keys) from the source application that is exposed to the users by the application interface. These sequence IDs are then adopted by the users as business keys. OrderID is an example from the Mr DVD example. OrderID is actually a system identifier (primary key) produced by the DVD POS point-of-sale system but, because the business uses it when printing invoices, searching for transactions and suchlike, it qualifies as a business key. Source systems occasionally fail to make the business key unique. In such cases the business key becomes a composite key comprised both the “natural key” and the source system primary key. However, this option is not recommended and should be prevented where possible (Linstedt & Graziano, 2010:45).

Business keys are the most stable piece of data in a business. Although the descriptive information about business keys may change, the keys seldom do. As an example from Mr DVD, consider its employees. The business key that uniquely identifies an employee is the employee number (EmployeeNo). However, an employee’s last name may change when she marries. This employee may move to another address, be assigned another telephone number and, in extreme cases, her gender may even change. However, her EmployeeNo will always remain stable and be a unique identifier of that particular employee. This property of business keys provides extreme stability to hubs and, in turn, to the data vault (Linstedt & Graziano, 2010:46).

Business keys should be identical across all lines of a business. However, this is often not the case. This can be attributed to the use of different applications across the lines of business, typing errors, and so on. Figure 3.15 below illustrates an example of a customer business key that changes across different lines of business. The customer code, abc345, changes from the marketing department to the sales department and again to the finance department. However, this is always bad for business as it makes it difficult to integrate data across systems at the different departments. This issue may be addressed in the data vault by a component known as the same-as link. Same-as links will be discussed later.

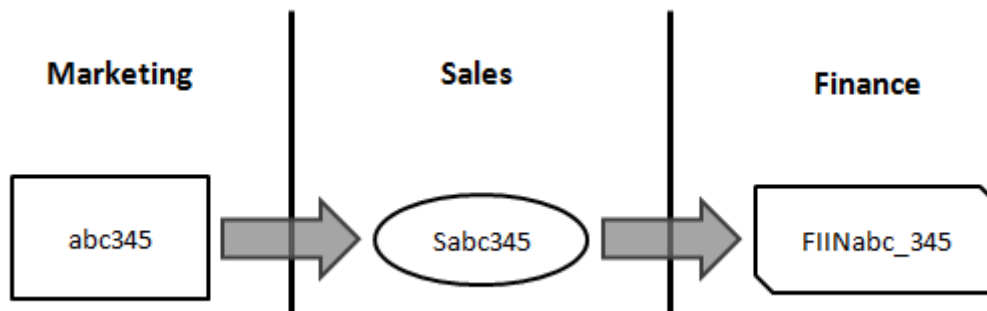


**Figure 3.15 Changing business keys across lines of business**

Linstedt (2010e) distinguishes between the semantic meanings of business keys. Identical business keys arriving in the data vault from different source systems can either have the same semantic meaning or a different semantic meaning. For example, a production company business key of PAR might arrive from two different sources. When the keys are an identifier for the same production company, Paramount Pictures, the keys are said to have the same semantic meaning. However, if the business key of PAR identifies different production companies in each of the source systems, the keys are said to have a different semantic meaning. The term “collisions” is used when keys with different semantic meanings arrive in the data vault. Collisions are handled by combining the natural key (PAR) with the primary key from the source system into a composite business key for the hub.

Linstedt and Graziano (2010:54) describe a five-step process for modelling data vault hubs:

1. Identify the business keys. This is achieved by talking to the staff members of the business, observing how they interact with software applications, looking at reports, documents, invoices, and so on.
2. Validate these business keys. Business keys should always be verified by the business users. A sanity check on business keys should be conducted to determine whether they are implemented as unique indexes in source systems.
3. Check business keys against multiple source systems and ensure that they have the same semantic meaning in all systems.
4. Build the hub that contains the business key, a sequence number, record source, and load date.
5. Finally, validate the hub by reconstructing the business keys for each of the sources that feed the hub.

This concludes the section on hub entities. The next section will discuss satellite entities – the second component of the data vault.

### 3.6.2 Satellite entities

Linstedt and Graziano (2010:108) describe satellite entities as the warehousing portion of the data vault model. Satellite entities provide context to hubs and links by housing detailed descriptive data about these entities over time. New records about hubs/links can only be loaded into satellites only if at least one of the descriptive fields in the satellite has changed. The structure of satellites and an example of a satellite are presented in figure 3.16 below. Satellites have a composite primary key that comprises the record load date timestamp and the primary key of the hub/link to which it provides context. The only foreign key found in satellites is the primary keys of its parent table (either a hub or a link).

| SAT_Entity           |                                                                         |
|----------------------|-------------------------------------------------------------------------|
| <b>PK,FK1<br/>PK</b> | <b><u>HUB_SQN</u><br/><u>SAT_LEDTS</u></b>                              |
|                      | SAT_Descriptive_Info1<br>SAT_Descriptive_Info2<br>SAT_LEDTS<br>SAT_RSRC |

| SAT_DVD              |                                                                         |
|----------------------|-------------------------------------------------------------------------|
| <b>PK,FK1<br/>PK</b> | <b><u>DVD_SQN</u><br/><u>LEDTS</u></b>                                  |
|                      | TitleSerial<br>ReleaseDate<br>RunningTime<br>UnitPrice<br>LEDTS<br>RSRC |

**Figure 3.16 Satellite entity structure and an example of a satellite**

The load end date timestamp (LEDTS) is mandatory for satellites. Each record in the satellite contains descriptive information of its parent hub/link. If any aspect of the descriptive information changes, a new record is inserted into the satellite and the previous record is end-dated. This, in turn, creates a mechanism with which to track changes in descriptive data over time. There should always be one record per hub/link sequence number only that is not end-dated (Linstedt & Graziano, 2010:108).

Table 3.52 below presents an example of changing data in a satellite over time from the Mr DVD example. Note that DVD\_SQN 2 appears twice in the table. The DVD initially had a unit price of R180.85 but, on 2010/10/11, the unit price decreased to R160.95 and, thus, the previous record for that DVD was end-dated. End-dating the previous record provides that



record with an effective time span from 2010/10/05 to 2010/10/10. Note also that all current records have a LEDTS of null.

**Table 3.52 Example of changing data in a satellite table**

| DVD_SQN | LDTs       | TitleSerial | UnitPrice | RunningTime | LEDTS      | RSRC    |
|---------|------------|-------------|-----------|-------------|------------|---------|
| 1       | 2010/10/05 | CLIFFHAN00  | 160.95    | 113         | NULL       | DVD.POS |
| 2       | 2010/10/05 | DRDOLITT00  | 180.85    | 85          | 2010/10/10 | DVD.POS |
| 3       | 2010/10/05 | EXTRAORD00  | 210.95    | 106         | NULL       | DVD.POS |
| 2       | 2010/10/11 | DRDOLITT00  | 160.95    | 85          | NULL       | DVD.POS |

All the descriptive information for each hub/link may be stored in a single satellite entity. However, this will almost always cause problems if there are a large number of descriptive fields. For example, assume the customer satellite in the Mr DVD example had fifty descriptive fields. If a single one only of these changes, all the fields (including the forty-nine fields that have not changed) have to be duplicated in a new row. This would soon cause storage problems in systems in which large volumes of data exist. In order to address this concern, satellites are usually normalised by type. Two major type categories exist, namely:

1. source system
2. rate of change.

Normalising the satellite by source system requires one satellite for each source system. Assume that, in addition to the DVD POS system, Mr DVD also uses a Pastel Payroll system. As illustrated in figure 3.17 below, both of these systems contain descriptive information for the employee hub. In the example the employee satellite is split by the two source systems that feed it; namely, sales (DVD POS) and HR (Pastel Payroll). Each of these source systems has different fields. The benefit of this source system technique is that there is no need to integrate information. In other words, the name (HR) and first and last names (sales) do not have to be integrated into one field called employee name. Another benefit of the technique is the fact that queries from each line of business may use the satellite from the source system that they prefer. For example, sales people may use SAT\_Emp\_Sales while HR staff may use SAT\_Emp\_HR. It is important to note that the satellites in the example describe information pertaining to HUB\_Emp and that each employee has one key in the employee hub entity only.

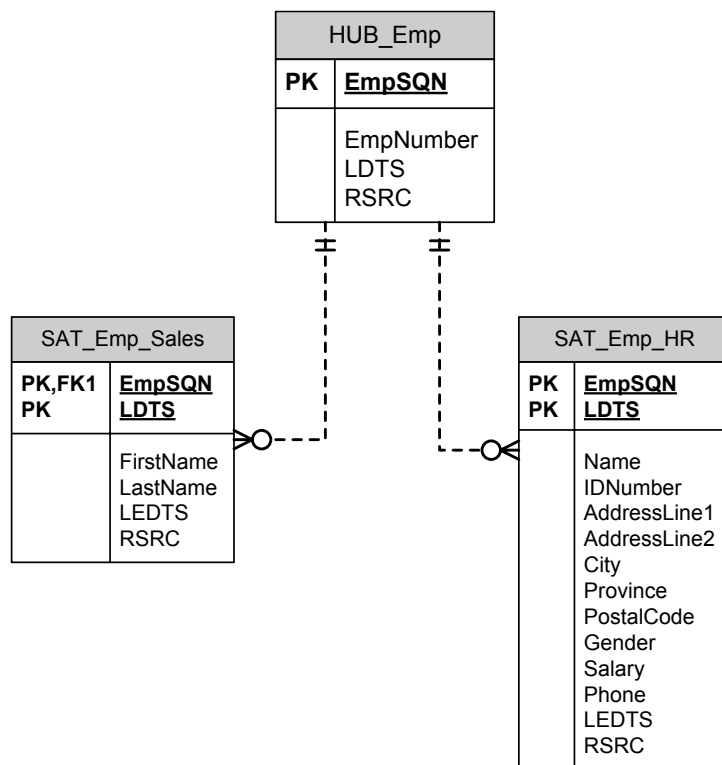


**Figure 3.17 Satellite normalisation by source system (Linstedt & Graziano, 2010:117)**

Fields in satellites may also be normalised according to rate of change. Some attributes in a satellite will change more frequently than others. If these attributes are in the same satellite, even those attributes with a slower rate of change will be duplicated every time an attribute with a faster rate of change changes. Table 3.53 below illustrates the process with an example from the Mr DVD DVD satellite. Note that the unit price of the Rocky DVD changed three times while the other attributes remained unchanged. However, because the unchanged attributes were in the same satellite table as the unit price they, too, were duplicated every time the unit price changed.

**Table 3.53 Illustrating rate of change in satellite table**

| DVD_SQN | LDTS       | TitleSerial | UnitPrice | RunningTime | LEDTS      | RSRC    |
|---------|------------|-------------|-----------|-------------|------------|---------|
| 5       | 2010/10/05 | ROCKY00000  | 160.95    | 119         | 2010/10/12 | DVD.POS |
| 5       | 2010/10/13 | ROCKY00000  | 170.95    | 119         | 2010/10/17 | DVD.POS |
| 5       | 2010/10/18 | ROCKY00000  | 150.95    | 119         | 2010/10/25 | DVD.POS |
| 5       | 2010/10/26 | ROCKY00000  | 120.75    | 119         | NULL       | DVD.POS |

Table 3.54 and table 3.55 below illustrate what would happen if the attributes were normalised into different satellites according to rate of change. The TitleSerial and RunningTime attributes would not be duplicated when the UnitPrice attribute changed and, thus, significantly less storage space would be required. Note that there are certain storage overheads in the auditing attributes that will always be duplicated. However, the storage space that will be saved will increase dramatically as more descriptive attributes are added to the satellite entity.

**Table 3.54 Normalising attributes by rate of change**

| DVD_SQN | LDTs       | TitleSerial | RunningTime | LEDTS | RSRC    |
|---------|------------|-------------|-------------|-------|---------|
| 5       | 2010/10/05 | ROCKY00000  | 119         | NULL  | DVD.POS |

**Table 3.55 Normalising attributes by rates of change**

| DVD_SQN | LDTs       | UnitPrice | LEDTS      | RSRC    |
|---------|------------|-----------|------------|---------|
| 5       | 2010/10/05 | 160.95    | 2010/10/12 | DVD.POS |
| 5       | 2010/10/13 | 170.95    | 2010/10/17 | DVD.POS |
| 5       | 2010/10/18 | 150.95    | 2010/10/25 | DVD.POS |
| 5       | 2010/10/26 | 120.75    | NULL       | DVD.POS |

The researcher is of the opinion that normalising satellites by a combination of source system and rate of change would be the most effective approach.

Link entities constitute the third major component of the data vault.

### 3.6.3 Link entities

If hub entities are the components that provide stability to the data vault model, then link entities are the components that provide extreme flexibility to the data vault. Links are tables that are used to define relationships between the business keys/hubs. Linstedt and Graziano (2010:57) describe links as “an intersection of business keys”.

At the very least link entities always possess the following attributes – see figure 3.18 below:

- sequence number
- foreign keys to two or more hubs

- load date
- record source.

The hub foreign keys in the link entity always form a unique index or alternate key.

| LINK |                                                                          | LNK_DVD_Actor_RoleType |                                                      |
|------|--------------------------------------------------------------------------|------------------------|------------------------------------------------------|
| PK   | <u>Sequence_Number</u>                                                   | PK                     | <u>DVDActorRoleType_SQN</u>                          |
|      | Business_Key_1 (FK)<br>Business_Key_2 (FK)<br>Load_Date<br>Record_Source | FK2<br>FK1<br>FK3      | DVD_SQN<br>Actor_SQN<br>RoleType_SQN<br>LDTS<br>RSRC |

**Figure 3.18 Mandatory components of link entities and LNK\_DVD\_Actor\_RoleType example**

The flexibility of the links lies in the fact that links always represent many-to-many relationships between business keys as opposed to one-to-many or many-to-one relationships (Linstedt & Graziano, 2010:57). This is best illustrated in the following example and is depicted in figure 3.19 below. Currently, a business rule of Mr DVD states that a DVD may belong to one, and only one, category although several DVDs may be assigned to each category. This, in turn, implies a one-to-many relationship between DVDs and categories. If the relationships between HUB\_DVD and HUB\_Cat were modelled as one-to-many, data could be loaded into the data vault. However, it would not be possible to load the data into the data vault if the business rule changed to allow DVDs to belong to multiple categories (many-to-many relationship) without making a change to both the data vault as well as to the loading routines that load these tables. If link entities are used between hubs to facilitate many-to-many relationships between the hubs, all relationships can be accommodated in the data vault (see figure 3.20 below) without making changes to either the model or loading routines.

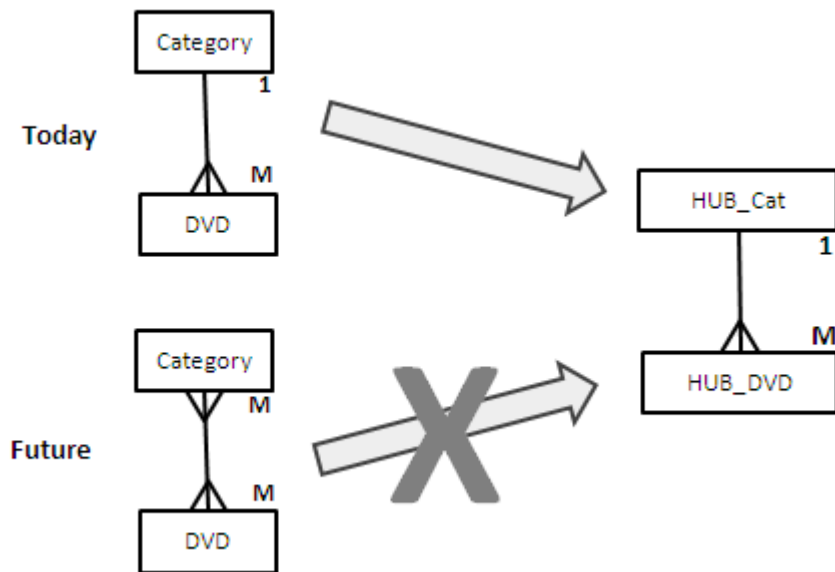


**Figure 3.19 Enforcing one-to-many or many-to-one relationships between hubs**

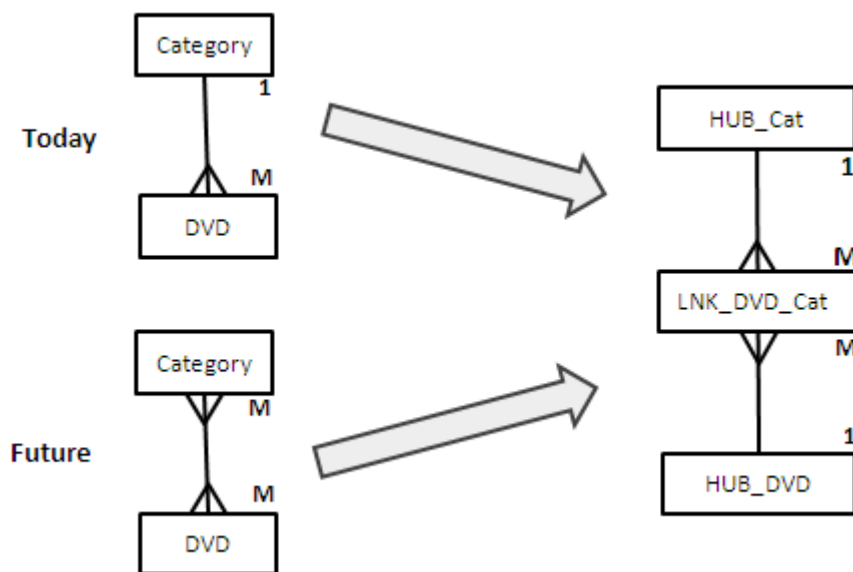


**Figure 3.20 Implementing many-to-many relationships between hubs through links**

Linstedt and Graziano (2010:74) group links into three different categories:

- hierarchical links
- same-as links
- transactional links.

Hierarchical links are used to capture the hierarchical relationships between the elements (or records) of a single hub entity. An example of a hierarchical relationship from the Mr DVD example is the recursive relationship between employees and managers. Figure 3.21 below depicts the relationship, while figure 3.22 presents the data vault model used to implement the hierarchy.

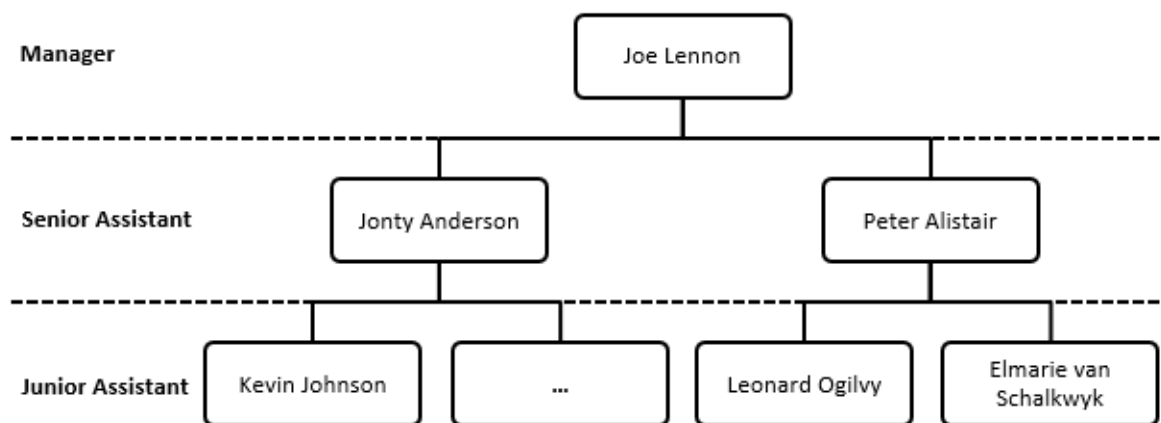


**Figure 3.21 Example of mine section hierarchy**

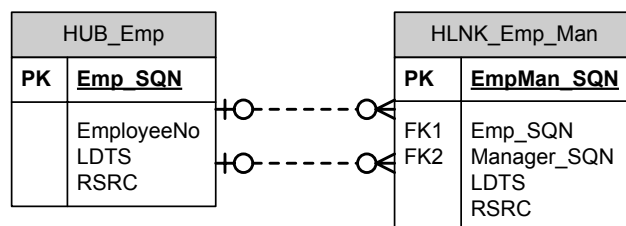


**Figure 3.22 Example hierarchical link data vault model**

Same-as links are used primarily in instances in which there are collisions in the data from different source systems arriving in the data vault (Linstedt & Graziano, 2010:76). Collisions

can be defined as either different pieces of data with the same semantic meaning or the same piece of data having different meanings in different source systems. The name of a movie production company which is spelt differently across different source systems is one example of this. These collisions occur mainly as a result of either typing errors or a lack of master data management. Figure 3.23 below presents an example of a collision on the Mr DVD production company name which is the business key of HUB\_ProdCom. Different sources have provided different spellings (or forms) for the same supplier. In this case, all forms of the supplier business keys are recorded in HUB\_ProdCom. However, one system is selected as the master source system and, in that system, the name of the production company is “20th Century Fox”. All other forms of the name are mapped to the main form in the same-as link.

Figure 3.24 below present examples of the hub and same-as link (HUB\_ProdCom and SALNK\_ProdCom) while table 3.56 and table 3.57 below present the data that is required in HUB\_ProdCom and SALNK\_ProdCom respectively for the purposes of this example. Note that the value for the record source column of the same-as link is set to “SYSTEM.GENERATED”. This is as a result of the fact that the business rule that maps the alternative production company business keys to the master production company business key is implemented in the data vault although the situation would be different if the mapping existed somewhere in a source system.

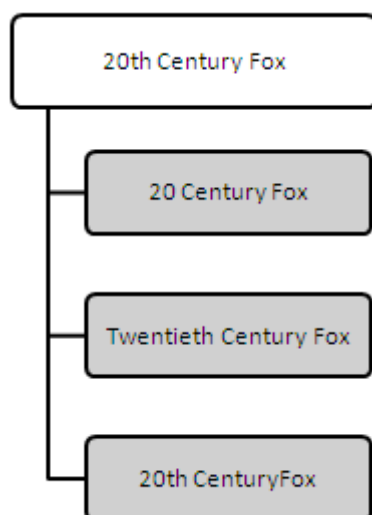


**Figure 3.23 Example of collisions on a production company business key**

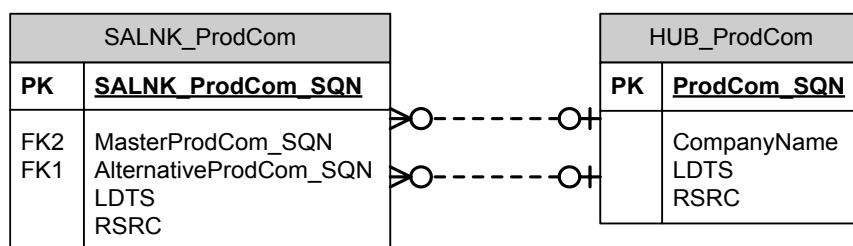


**Figure 3.24 Example of a data vault model for handling collisions on the production company**

**Table 3.56 Example of HUB\_ProdCom for handling collisions on the production company**

| ProdCom_SQN | CompanyName           | LDTS       | RSRC           |
|-------------|-----------------------|------------|----------------|
| 1           | 20th Century Fox      | 2010/10/04 | DVD.POS        |
| 8           | 20 Century Fox        | 2010/10/05 | DVD.POS        |
| 9           | Twentieth Century Fox | 2010/10/16 | PASTEL.FINANCE |
| 10          | 20th CenturyFox       | 2010/10/24 | PASTEL.FINANCE |

**Table 3.57 Example of same-as link for handling collisions on the production company**

| SALNK_ProdCom_SQN | Master ProdCom_SQN | Alternative ProdCom_SQN | LDTS       | RSRC             |
|-------------------|--------------------|-------------------------|------------|------------------|
| 10                | 1                  | 1                       | 2010/10/04 | SYSTEM.GENERATED |
| 11                | 1                  | 8                       | 2010/10/05 | SYSTEM.GENERATED |
| 12                | 1                  | 9                       | 2010/10/16 | SYSTEM.GENERATED |
| 13                | 1                  | 10                      | 2010/10/24 | SYSTEM.GENERATED |

Transactional links entities are link entities that record transactions. However, these transactions are nothing more than relationships between business keys. For example, as depicted in figure 3.25 below, a Mr DVD sales transaction may be defined as a relationship between DVDs and orders (for this particular example, each line item is considered as a transaction). Links usually record the association between business keys only, but no other information related to the association. The information related to the association between DVDs and orders comprises the line number and quantity which would be stored in a satellite.



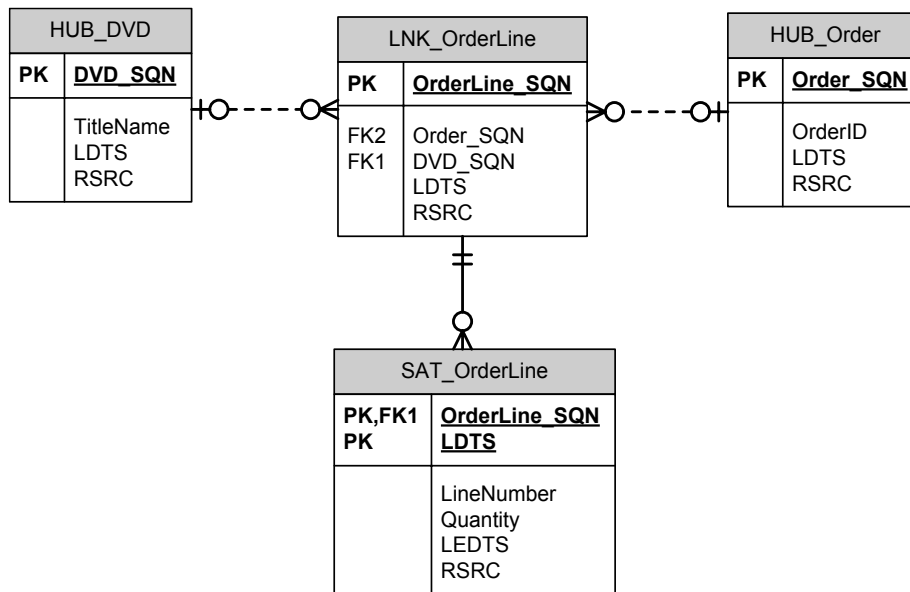


**Figure 3.25 Transactional link data vault model example**

Linstedt and Graziano (2010:80) define transactions as events that are recorded in the data vault and that cannot legally be changed. In other words, it is not possible for there to be different versions of the descriptive information pertaining to a transaction. Recall that satellites have the ability to record changes in data values over time. Accordingly, the satellite in figure 3.25 above would need to be modified so that it would not allow for different versions of the descriptive information. The model may be changed in one of two ways (Linstedt & Graziano, 2010:80). The first alternative, referred to by the researcher as type 1 transactional links, involves removing the load end date timestamp from the satellite, and also removing the load date timestamp from the satellite's primary key. This, in turn, changes the relationship between the link and satellite entity to one-to-one, thus allowing the satellite to store one record for each record in the link entity only. The change to the satellite is depicted in figure 3.26 below. The second alternative, referred to by the researcher as type 2 transactional links, involves denormalising the model and collapsing the information of the satellite into the link. The second alternative is depicted in figure 3.27 below.

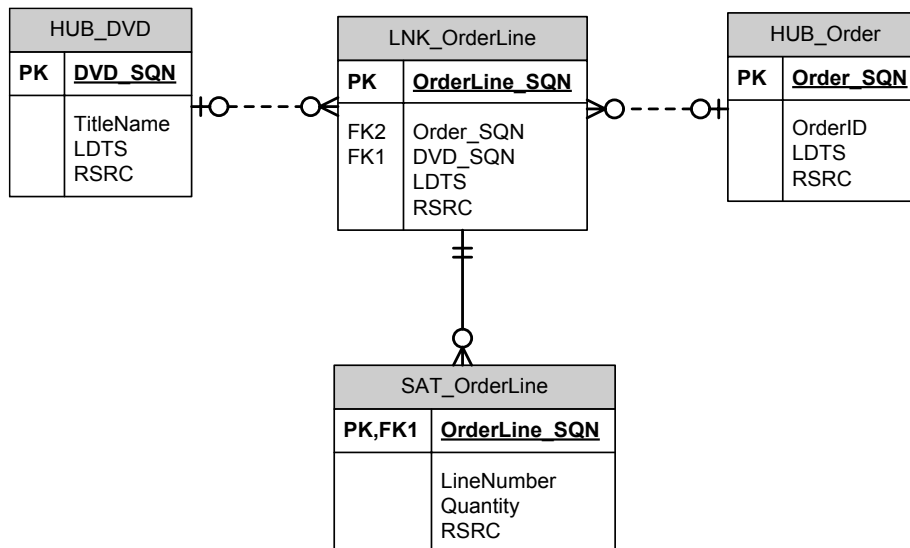


**Figure 3.26 Changes to satellite for transactional links**

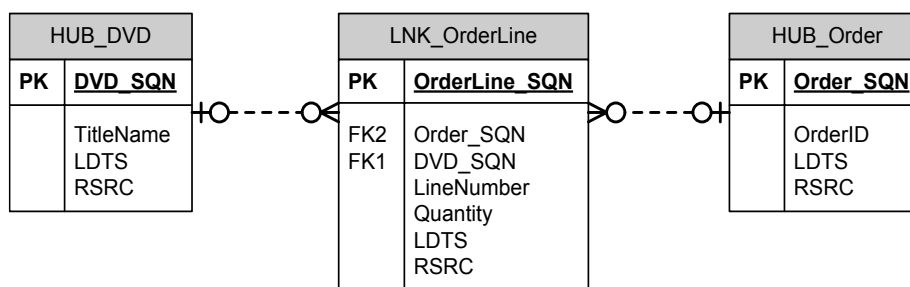


**Figure 3.27 Collapsing the satellite into the link for transactional links**

When collapsing the satellite information into the transactional link, it is essential that the data modeller take into account both the arrival time of the data in the data vault and the width of the data in the link entity. Transactional links should never be updated. In addition, there should be inserts on the link table only. It may happen in real-time environments that the transaction and the descriptive information for the transaction do not arrive at the same time. In this case, it would not be possible to collapse the satellite into the link, as not all the information that the collapsed transactional link requires when the first piece of data for the transaction arrives will be available (Linstedt, 2010a). Another scenario that would render the collapsed link a less preferable alternative is if the row becomes too wide. The number of records that can be read by the DBMS in a single I/O operation may be greatly reduced and this, in turn, will have a negative impact on performance (Linstedt & Graziano, 2010:82).

### 3.6.4 Link-to-link relationships

Link entities are not limited to modelling associations between hubs but are also used to model relationships to other links. Linstedt and Graziano (2010:71) refer to these relationships as link-to-link or parent-child relationships. The concept is illustrated by the Mr DVD example depicted in figure 3.28 below. LNK\_Order models the relationship between customers, employees and order line items. Each order contains multiple line items while each line item on an invoice relates to a specific DVD. This is modelled by the LNK\_OrderLine.

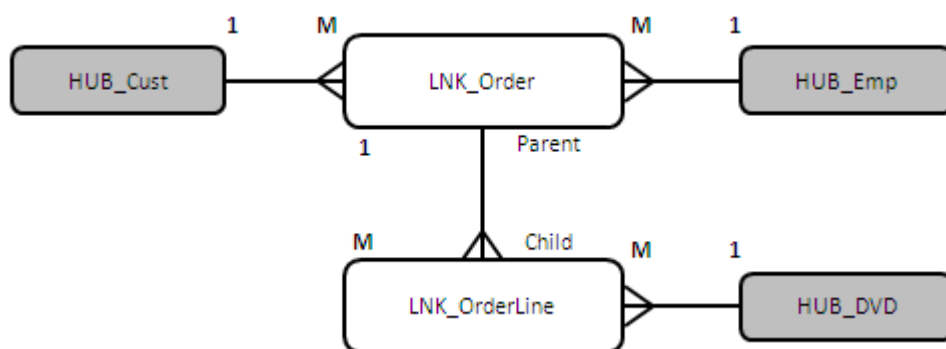


**Figure 3.28 Link-to-link (parent-child) relationship example for logical data vault model**

The logical model depicted in figure 3.28 above is correct. However, it should be treated as a logical model only and it should be physically implemented differently from the logical model. The biggest problem with link-to-link relationships is that they increase the complexity of the loading processes and queries (Linstedt & Graziano, 2010:72). However, these complexities can be avoided by removing the link-to-link relationship in the physical data model. This is a two-step process involving the following:

1. Denormalising the links by including all the hub keys from the parent link to the child link.
2. Removing the relationship between the links.

This denormalised model and the removed parent-child relationship are illustrated in figure 3.29 below. It is important to change the physical data model only and not the logical model, as it is not easy to identify parent-child relationships once the model has been denormalised.

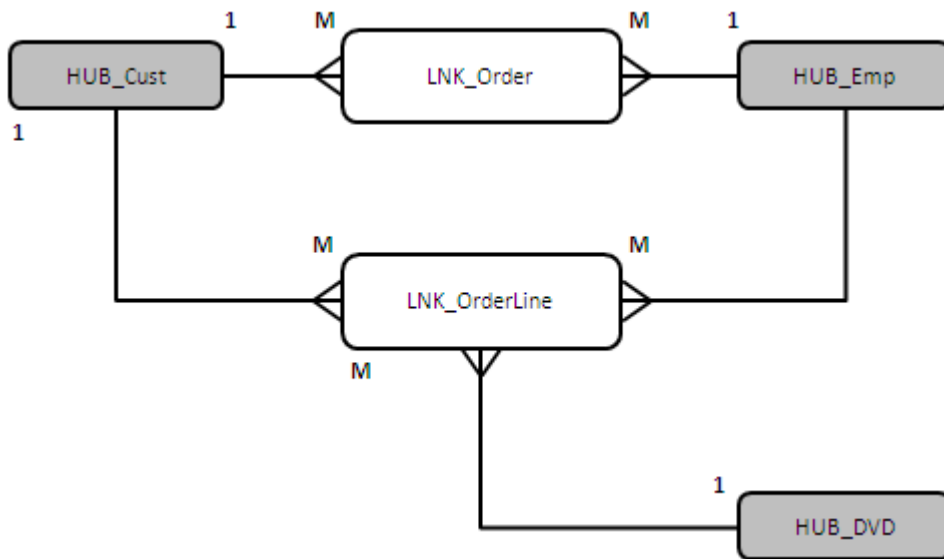


**Figure 3.29 Denormalised link-to-link (parent-child) relationships**

### 3.6.5 Bridge tables

Bridge tables are tables that are used to alleviate the performance issues caused by the large number of joins that result from data vault models (Linstedt & Graziano, 2010:140). According to Linstedt and Graziano (2010:139), bridge tables are not part of the data vault architecture. However, bridge tables improve query performance by reducing the number of joins required for any particular query. This is achieved through the migration of several hub entity foreign keys from two or more link entities into a single table. The structure of a bridge table and an example from Mr DVD are presented in figure 3.30 below.

| Bridge |                                                     |
|--------|-----------------------------------------------------|
| PK     | <u>Bridge_SQN</u>                                   |
|        | HUB_1_SQN<br>HUB_2_SQN<br>HUB_N_SQN<br>LDTS<br>RSRC |

| BRI_DVD_Cat_ProdCom |                                                   |
|---------------------|---------------------------------------------------|
| PK                  | <u>Bridge_SQN</u>                                 |
|                     | DVD_SQN<br>Cat_SQN<br>ProdCom_SQN<br>LDTS<br>RSRC |

**Figure 3.30 Bridge table structure and example**

In order to clarify the use of bridge tables, consider the following example from Mr DVD. Figure 3.31 below presents a portion of the Mr DVD data vault model. Joe Lennon wants a report that shows the DVDs in each category as well as the production company that

produced these DVDs. An example of this report is presented in table 3.58 below. Figure 3.32 below shows the SQL query to the data vault that will return the data required for Joe's report. Note that the query contains a four-way join to five tables.

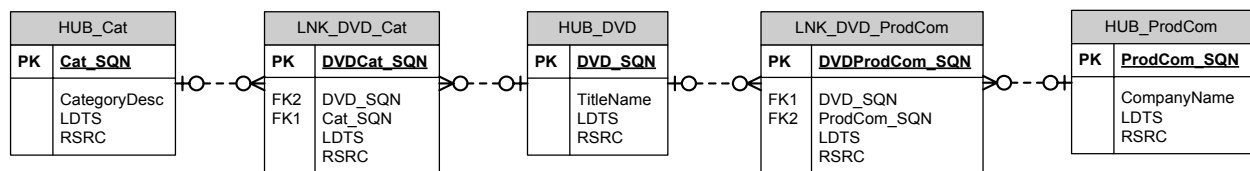


**Figure 3.31 HUB\_DVD, HUB\_Cat and HUB\_ProdCom with related link entities**

**Table 3.58 DVDs by category and production company**

| CategoryDesc | TitleName                      | CompanyName        |
|--------------|--------------------------------|--------------------|
| Action       | Cliffhanger                    | Sony Pictures      |
|              | Rocky                          | MGM Studios        |
|              | Wanted                         | Universal Studios  |
| Adventure    | Indiana Jones and the Lost Ark | Paramount Pictures |
| Animation    | Shrek                          | DreamWorks         |
| Comedy       | Dr. Dolittle                   | 20th Century Fox   |
|              | What Happens in Vegas          | 20th Century Fox   |
| Drama        | Extra Ordinary Measures        | Sony Pictures      |
| Thriller     | Salt                           | Columbia Pictures  |

```

SELECT
    hc.CategoryDesc,
    hdvd.TitleName,
    hpc.CompanyName
FROM
    dv.HUB_DVD hdvd JOIN
    dv.LNK_DVD_Cat ldvdc ON hdvd.DVD_SQN = ldvdc.DVD_SQN JOIN
    dv.HUB_Cat hc ON hc.Cat_SQN = ldvdc.Cat_SQN JOIN
    dv.LNK_DVD_ProdCom ldvdpc ON ldvdpc.DVD_SQN = hdvd.DVD_SQN JOIN
    dv.HUB_ProdCom hpc ON hpc.ProdCom_SQN = ldvdpc.ProdCom_SQN
ORDER BY
    1, 2

```

**Figure 3.32 SQL query to data vault without bridge table**

In order to reduce the number of joins and improve the performance of the query, a bridge table can be constructed. The data for the bridge table from Mr DVD is presented in table 3.59 below. The query that uses the bridge table and returns the same dataset as the query in figure 3.32 above is depicted in figure 3.33 below. Note that this query contains a three-way join to four tables instead of a four-way join to five tables. The number of tables in similar join operations (without the use of bridge tables) can be calculated as:

$$h + (h - 1) + s$$

Where  $h$  = number of hub entities involved in the query

$s$  = number of satellite entities involved in the query

The number of tables in similar queries that use a bridge table can be calculated as:

$$h + 1 + s$$

Where  $h$  = number of hub entities involved in the query

$s$  = number of satellite entities involved in the query

If the example query had required data from 20 hub entities and did not make use of a bridge table then the resultant query would contain a 38-way join to 39 tables. However, if the query did make use of a bridge table then the resultant query would contain a 20-way join to 21 tables only.

**Table 3.59 Sample bridge table data**

| BRI_DVD_Cat_PC | DVD_SQN | Cat_SQN | ProdCom_SQN | LDTs       | RSRC   |
|----------------|---------|---------|-------------|------------|--------|
| 1              | 62      | 15      | 59          | 2010/10/23 | SYSTEM |
| 2              | 63      | 18      | 54          | 2010/10/23 | SYSTEM |
| 3              | 64      | 19      | 59          | 2010/10/23 | SYSTEM |
| 4              | 65      | 16      | 58          | 2010/10/23 | SYSTEM |
| 5              | 66      | 15      | 57          | 2010/10/23 | SYSTEM |
| 6              | 67      | 20      | 55          | 2010/10/23 | SYSTEM |
| 7              | 68      | 17      | 56          | 2010/10/23 | SYSTEM |
| 8              | 69      | 15      | 60          | 2010/10/23 | SYSTEM |
| 9              | 70      | 18      | 54          | 2010/10/23 | SYSTEM |

```

SELECT
    hdvd.TitleName,
    hc.CategoryDesc,
    hpc.CompanyName
FROM
    dv.BridgeTest b JOIN
    dv.HUB_DVD hdvd ON b.DVD_SQN = hdvd.DVD_SQN JOIN
    dv.HUB_Cat hc ON b.Cat_SQN = hc.Cat_SQN JOIN
    dv.HUB_ProdCom hpc ON b.ProdCom_SQN = hpc.ProdCom_SQN

```

**Figure 3.33 SQL query utilising bridge table to reduce the number of joins**

### 3.6.6 Cross-reference tables

Cross-reference tables are another component of the data vault modelling architecture as defined by Linstedt (2010c). Cross-reference tables, also known as lookup tables, key resolution tables, or code description tables, contain codes and descriptions that are used frequently throughout the model. They are typically modelled as standalone tables in the data vault although the relationships to other entities are not modelled because this would complicate the model too much. Cross-reference tables can be modelled in one of two ways:

- as a single denormalised table (termed XREF tables) or
- as cross-reference hub and satellite tables (called XHUB and XSAT).

Modelling cross-reference tables as XREF tables is the simpler of the two options. However, modelling cross-reference tables as XHUB and XSAT tables allows the structure to store the history of code and description changes over time (Linstedt, 2010d). A date XREF table is an example of the first option. In this example, the date surrogate smart key would be the code.

The researcher recommends using the XREF tables only in instances in which the description for each code may logically never change (unless incorrectly loaded). An example of such an instance would be dates and descriptions. A date code of 2010/05/27 will always have a full date description of 27 May 2010 and this would never change.

### 3.6.7 Four-step (or seven-step) process to data vault modelling

According to Linstedt (2003), data vaults may be modelled by following the four-step process:

1. Model the hubs.
2. Model the links.
3. Model the satellites.
4. Normalise the satellites.

In order to model the hubs, the business keys need to be identified. Each business key will form a single hub entity. Hubs from the Mr DVD example include actor (HUB\_Actor), category (HUB\_Cat), customer (HUB\_Cust), DVD (HUB\_DVD), employee (HUB\_Emp), job title (HUB\_JobTitle), production company (HUB\_ProdCom) and actor role type (HUB\_Role\_Type).

The second step, modelling the links, involves the identification of the relationships between all the hubs. Links include transactional links, hierarchical links, and so on. Link entities from the Mr DVD example include the link between:

- DVDs and categories (LNK\_DVD\_Cat)
- DVDs and production companies (LNK\_DVD\_ProdCom)
- employees and job titles (LNK\_Emp\_JobTitle)
- DVDs, actors and role types (LNK\_DVD\_Actor\_RoleType)
- employees, customers and order line items (LNK\_Order)
- DVDs and orders (LNK\_OrderLine).

The third step in the process involves modelling the satellites. Satellites are easily identified as any piece of information that provides descriptive information to either hub entities or link entities. Satellites from the Mr DVD example include information about actors (SAT\_Actor), categories (SAT\_Cat), customers (SAT\_Cust), DVDs (SAT\_DVD), employees (SAT\_Emp), job titles (SAT\_JobTitle), production companies (SAT\_ProdCom), actor role types (SAT\_RoleType), orders (SAT\_Order) and order line items (SAT\_OrderLine).

Finally, satellite entities are normalised according to rate of change and source system. In view of the fact that the data for Mr DVD comes from a single source system, satellites are



normalised by rate of change only. The only satellite that contains descriptive information with different rates of change is SAT\_Cust. The FirstName, LastName and Gender fields are relatively stable. However, address-related fields and the customer telephone numbers may change fairly frequently. Accordingly, SAT\_Cust is normalised into two satellite entities, namely, SAT\_Cust\_Name\_Gender and SAT\_Cust\_Addr\_Phone.

According to the researcher, this process is missing three steps. These missing steps were identified after the researcher had followed the step-by-step process and, at the end, was still able to identify some changes or additions to the model that were required. The steps include the following:

1. Flatten link-to-link relationships. As explained in section 3.6.4 (Link-to-link relationships), link-to-link relationships should be flattened in order to improve data loading and query performance.
2. Transactional links should be transformed to either type 1 or type 2 transactional links. Recall from section 3.6.3 (Link entities) that this transformation is required in order to ensure that it is not possible to make any changes to the transactional data.
3. Add cross-reference tables. Although these tables do not have modelled relationships to the other entities, they are still required as standalone entities on the model. This step would add the date cross-reference table (XREF\_Date) to the Mr DVD data vault model.

### **3.6.8 Data vault indexing strategies**

As an initial indexing strategy for a data vault, Evers (2011) recommends a unique clustered index for hub entities comprising the hub surrogate key as well as a unique clustered index on the business key field. Unique clustered indexes are recommended on link sequence numbers with non-unique non-clustered indexes on each of the hub entity foreign keys in the link entity. As regards the satellite entities, Evers recommends a unique clustered index on the primary key of the satellite, which is always the hub foreign key, followed by the satellite load date time stamp.

## 3.7 Comparison between dimensional and data vault modelling techniques

This section focuses on the similarities and differences between the dimensional and data vault modelling techniques, as well as a preliminary comparison in terms of flexibility and query performance. The comparison is theoretical and not based on empirical results of experiments. The empirical results of the experiments are presented in chapter 4.

### 3.7.1 Smart keys

Both Kimball *et al.* (2008:253) and Linstedt (2010f) warn against the use of smart keys although they do, allow the use of smart keys under specific circumstances. Kimball *et al.* (2008:254) allow the use of smart keys only when they are used as date keys. For example, a date key of 20100703 may be used as the surrogate key for 3 July 2010 in the date dimension. In general, Linstedt advises against the use of smart keys, but recommends that, if smart keys are used, they should always fulfil with the following conditions (2010f):

- The keys are recomputed at any point in time.
- The keys should always be unique.

The first condition merely requires a rule in terms of which the surrogate key may be calculated. For example, the rule that defines the smart key for a customer states that the first three characters of the last name and the first two characters of the first name should be used as a key. However, this rule would not necessarily always cater for the second condition. Consider two different customers, namely, Joe van der Walt and Jo-Anne van Zyl. When applying the example rule to these two customers, both will generate a smart key of vanJo and, thus, the key is clearly not unique.

### 3.7.2 Similarities between fact tables and transactional links

There is also a very clear similarity between the fact tables in dimensional modelling and type 2 transactional links in the data vault. As an example, consider figure 3.34 below that depicts the structure of a fact table on the left and a type 2 transactional link on the right. The similarities are listed below:

- Both tables contain similar transactional measurements, for example quantity and revenue/amount.

- Both tables contain foreign keys that join to customer, employee and DVD information.

The OrderDate field in LNK\_OrderLine can be used to look up descriptive date information in the date cross reference table. The OrderDate field in LNK\_OrderLine can be compared with the Date\_SK field in Fact\_Sales.

The only difference between the fact table and the type 2 transactional link table are the audit information and the primary surrogate key in the type 2 transactional link. Despite the fact that Kimball *et al.* (2008:377) do not consider the audit information as mandatory, it is allowed in the dimensional model. There are also instances in which a surrogate key in a fact table may be used as the primary key for a fact table (Becker, 2006).

| Fact_Sales                                         |                                                                                                  |
|----------------------------------------------------|--------------------------------------------------------------------------------------------------|
| PK,FK2<br>PK,FK1<br>PK,FK4,FK6<br>PK,FK3<br>PK,FK5 | <u>Date_SK</u><br><u>Customer_SK</u><br><u>Employee_SK</u><br><u>DVD_SK</u><br><u>ProdCom_SK</u> |
|                                                    | OrderID<br>Quantity<br>Revenue                                                                   |

| LNK_OrderLine     |                                                                                                          |
|-------------------|----------------------------------------------------------------------------------------------------------|
| PK                | <u>OrderLine_SQN</u>                                                                                     |
| FK2<br>FK1<br>FK3 | DVD_SQN<br>Cust_SQN<br>Emp_SQN<br>OrderDate<br>LineNumber<br>Quantity<br>OrderLineAmount<br>LDTS<br>RSRC |

**Figure 3.34 Example fact table (left) and type 2 transactional link (right)**

### 3.7.3 Similarities between slowly changing dimensions and satellite entities

Data vault satellite entities and dimensional modelling type 2 slowly changing dimensions are extremely similar. This is best illustrated in figure 3.35 below. Both of these entities have a similar purpose, namely, to track changes to data over time. In both cases this is achieved through a start date (load date timestamp) and an end date (load end date timestamp). However, there is the difference between the two entities in that a single satellite will usually track changes over less fields than a type 2 slowly changing dimension. Satellites are generally normalised by rate of change and source system whereas type 2 slowly changing dimensions are not generally normalised. Accordingly, multiple satellites are required to track changes to the same number of fields that a single type 2 slowly changing dimension would do. This may, however, be of concern with type 2 slowly changing dimensions. For example, if a type 2 slowly changing dimension contains 40 descriptive fields and a single

one of these fields changes, a new record is created in the table with this new record duplicates the information of the 39 other fields that did not change. This may, in turn, result in a rapidly growing table with large amounts of duplicated data.

| SAT_Entity |                                                                         | Type_2_SCD |                                                                                       |
|------------|-------------------------------------------------------------------------|------------|---------------------------------------------------------------------------------------|
| <b>PK</b>  | <b><u>HUB_SQN</u></b>                                                   | <b>PK</b>  | <b><u>Dim_SK</u></b>                                                                  |
| <b>PK</b>  | <b><u>SAT_LDTS</u></b>                                                  |            |                                                                                       |
|            | SAT_Descriptive_Info1<br>SAT_Descriptive_Info2<br>SAT_LEDTS<br>SAT_RSRC |            | Descriptive_Info1<br>Descriptive_Info2<br>Start_Date<br>End_Date<br>Current_Indicator |

**Figure 3.35 Comparing satellite and type 2 slowly changing dimension**

### 3.7.4 The number of joins and expected query performance

Kimball *et al.* (2008:237) argue that one of the key benefits of the dimensional model is the enhanced query performance as a result of the reduced number of joins between tables (when compared to 3NF data models). The data vault model always has many more joins than a dimensional data model (even when compared to 3NF data models). As proof of this statement, consider the example presented in figure 3.36 and figure 3.37 below. Both data models contain descriptive information about DVDs. Figure 3.36 contains the DVD dimension as per the dimensional modelling technique while figure 3.37 contains the same information modelled according to the data vault modelling technique. However, obtaining DVD-related information from the dimensional model in this example would require no table joins although obtaining exactly the same information from the data vault model would require a four-way join between five tables.

| Dim_DVD   |                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------|
| <b>PK</b> | <b><u>DVD_SK</u></b>                                                                                       |
|           | <b>TitleSerial</b><br>TitleName<br>CategoryCode<br>CategoryDesc<br>ReleaseDate<br>RunningTime<br>UnitPrice |

**Figure 3.36 DVD dimension containing DVD related information**

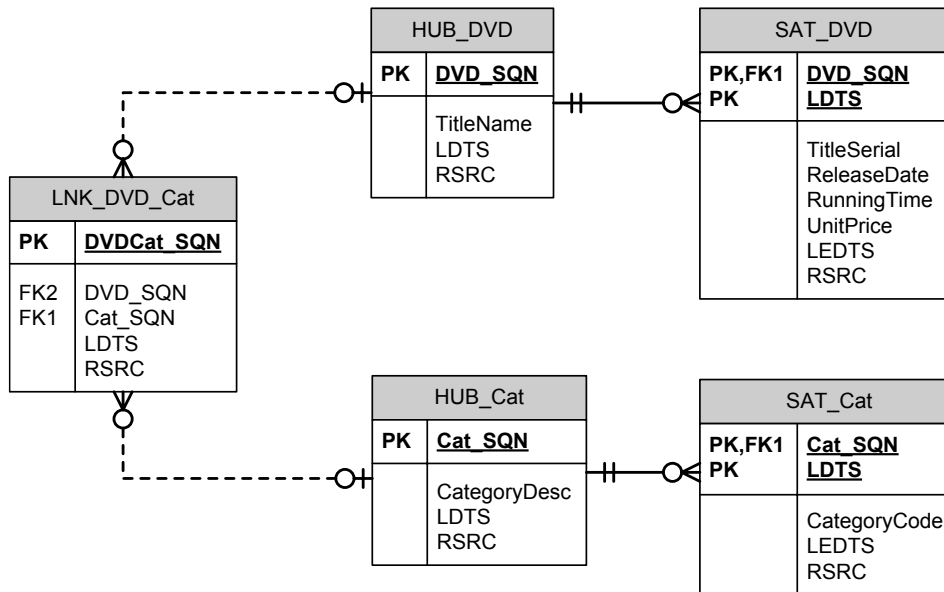


**Figure 3.37 Data vault model containing DVD related information**

The number of joins in a query has a direct impact on query performance (Kimball *et al.*, 2008:237; Linstedt, 2004). It may, thus, be deduced that query performance on a data vault model will be inferior when compared to that of a dimensional model. Linstedt (2004) argues that technology is to blame for the poor query performance on join operations and that the design of the data model should not be limited by technology, but that it should rather focus on modelling the business processes. However, the researcher is not completely convinced by this argument. Although the researcher does agree with the fact that database management technology is to blame for the poor performance on join operations, he does not think it is wise to ignore join performance completely. Any database design should be able to deal with the limitations of technology. Linstedt (2010b) defends the design, stating that it performs extremely well on massively parallel processing (MPP) databases and that, on MPP systems, queries on the data vault may even perform faster than dimensional models. This is as a result of the fact that the design allows multiple parts of a query to be executed in parallel.

### 3.7.5 Flexibility

Flexibility and tolerance to change are extremely important characteristics of any data model. If the data model is inflexible, changes in the business requirements would require a significant amount of reworking and maintenance to the existing tables in data models, the

loading processes that populate the models, as well as the queries that run off the models. An example is used to assist with the comparison of the two modelling techniques.

Consider the example where Mr DVD wants to add information on the suppliers of each DVD. Figure 3.38 below illustrates the data vault model before and after the change. Note that the model was extended to accommodate the change and that no changes were made to the existing tables. In order to accommodate the same change in the dimensional model, additional fields relating to the supplier would need to be added to the existing DVD dimension table. Figure 3.39 below illustrates the DVD dimension table before and after the change. Adding a separate supplier dimension would also be an option. However, even this would require the additional supplier surrogate key to be added to the fact tables, therefore altering the existing tables.

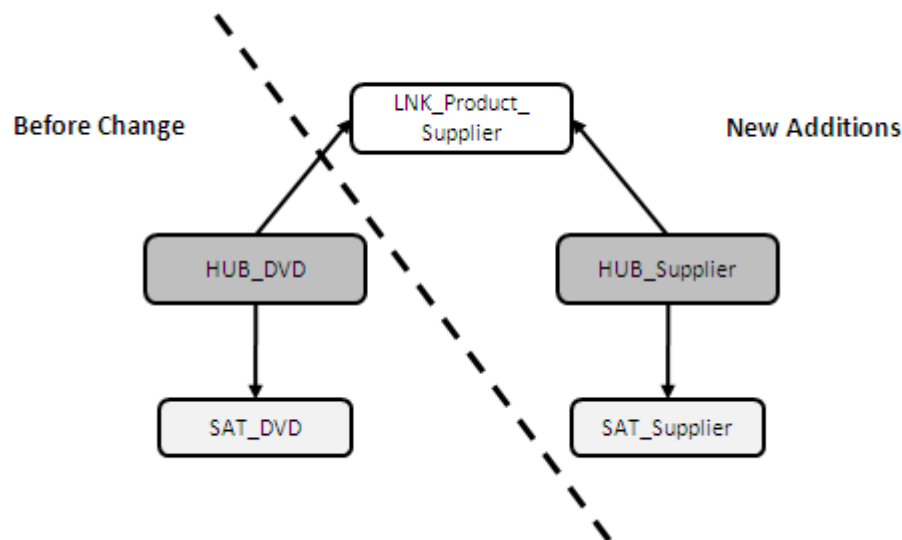


**Figure 3.38 Data vault model before and after change**

| Dim_DVD |                                                                                                     |
|---------|-----------------------------------------------------------------------------------------------------|
| PK      | <u>DVD_SK</u>                                                                                       |
|         | TitleSerial<br>TitleName<br>CategoryCode<br>CategoryDesc<br>ReleaseDate<br>RunningTime<br>UnitPrice |

| Dim_DVD |                                                                                                                                     |
|---------|-------------------------------------------------------------------------------------------------------------------------------------|
| PK      | <u>DVD_SK</u>                                                                                                                       |
|         | TitleSerial<br>TitleName<br>CategoryCode<br>CategoryDesc<br>ReleaseDate<br>RunningTime<br>UnitPrice<br>SupplierCode<br>SupplierDesc |

**Figure 3.39 Dim\_DVD before (left) and after (right) change**

Table 3.60 below summarises the impact that this change had on both data models. For this particular example it may be concluded that the data vault model was more resilient to change than the dimensional model. However, the researcher wishes to stress that this example used was a single, simplified example and that it is not possible to generalise the results to all examples. Chapter 4 will discuss the more rigorous testing that was conducted.

**Table 3.60 Summary of how the changed business rules impacted the data models**

|                          | Dimensional Model                                                                                                      | Data Vault Model                                                                                          |
|--------------------------|------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Existing table structure | The existing DVD dimension table changed                                                                               | Existing tables remain unchanged                                                                          |
| Loading routines         | Because the structure of the existing tables changed, the loading routines populating these tables will have to change | Because the structure of existing tables remained unchanged, no existing loading routines require changes |
| Existing queries         | Existing queries will continue to work without changes                                                                 | Existing queries will continue to work without changes                                                    |

### 3.8 Correlation between data warehouse modelling technique and data warehouse project success

The researcher tried to find previous research that showed a correlation between the success of the data warehouse and a specific data modelling technique as this would have enabled him to predict the outcome of the research. Unfortunately, no such research has been conducted to date. However, such research could create valuable knowledge and should be noted as an area for future work.

There are, however, multiple studies that have investigate the critical success factors for data warehousing projects, or else the main reasons why data warehousing projects fail, including reasons for success/failure that may be directly attributed to the data warehouse modelling technique (Adelman & Moss, 2000; Perkins, 2000). Adelman and Moss (2000) list unacceptable performance and the inability to expand as key reasons why data warehouse projects fail. They class performance issues into two categories, namely, query performance and loading performance. Both of these may be affected by the choice of data model with the scalability of the data model determining its ability to expand, while the flexibility of the data model determines the cost of the expansion.

Although these studies did not find a correlation between a specific data modelling technique and the success of a data warehouse project, it would appear that they do indicate that the data model does impact on the success of the project.

### **3.9 Summary**

The aim of this chapter was to introduce the reader to the two data warehouse modelling techniques that are relevant to this study, namely, dimensional modelling, a technique pioneered by Ralph Kimball (Kimball *et al.*, 2008:1), and data vault modelling, which was pioneered by Dan Linstedt (Linstedt, 2002).

Dimensional modelling and data vault modelling were explained through the use of the example of a DVD retailer. Dimensional modelling was discussed in terms of its core components, facts and dimensions. Certain advanced dimensional modelling concepts such as conformed dimensions, snowflaking, slowly changing dimensions, many-to-many dimension relationships and ragged/variable depth hierarchies were also discussed. As far as data vault modelling is concerned, the chapter discussed hub, satellite and link entities while certain advanced data vault modelling concepts, such as link-to-link relationships, bridge tables and cross-reference tables, were also discussed.

Finally, a theoretical comparison between dimensional modelling and data vault modelling was discussed. Although there are some similarities between these modelling techniques there are also significant differences between the two. These differences are expected to yield different results in experimentation with regards to data load performance, query performance and flexibility and tolerance to change.



The following chapter contains detailed descriptions of the empirical tests that were conducted as regards the comparison between dimensional modelling and data vault modelling.

# Chapter 4: Experiment

## 4.1 Introduction

This chapter discusses the execution and results of the experiments that was conducted as part of this study. The chapter starts with an overview in section 4.2 of each of the experiments which were conducted, followed by a description in section 4.3 of the environment in which the experiments were executed. Included is a motivation for the selection of a database management system (DBMS) and operating system (OS).

Several candidate data sources were identified with the TPC-E dataset being the dataset that was selected from these candidate data sources for the purposes of this experiment. This decision was based on a set of requirements that is discussed in section 4.4.1 while sections 4.4.3 and 4.4.4 describe the TPC-E data model in detail.

A set of fictitious user requirements is then described in section 4.5 and the resulting dimensional and data vault models that were created for the purposes of the experiment are discussed in sections 4.6.1 and 4.6.2 respectively. This section also provides detailed information on the way in which these models were designed and how they were implemented physically.

The chapter describes the execution of each of the experiments in section 4.7. This section includes a discussion on the formulation of the research question, the formulation of the hypotheses and the predicted results. This is followed by a description of the execution of the experiments and a presentation of the statistical analysis of the research results.

## 4.2 Overview of the experiments

This section provides a brief overview of each of the experiments that were conducted as part of this research study:

1. *Load performance.* Data warehouses contain large amounts of data. This data is loaded into the data warehouse through one of the following two processes, namely, historical or incremental loads. This experiment aims to compare the performance of

a dimensional model with the performance of a data vault model in terms of historical and incremental loads.

2. *Query performance*. The differences between the dimensional and data vault modelling techniques may result in differing levels of query performance. This experiment aims to compare the dimensional and data vault models in terms of query performance.
3. *Volume*. Each of the two target data models may have different data volume characteristics. This experiment will compare the two data models in terms of their resulting data volumes.
4. *Flexibility*. The final experiment aims to compare the two target data models in terms of their flexibility and resilience to change.

## 4.3 Testing server

The sections below describe the operating system and database management system that were used for the experiments and also how these were configured. Details of the hardware used are listed in Annexure O.

### 4.3.1 Database management system (DBMS)

One of the aims of the study was to compare the performance of a dimensionally modelled data warehouse with that of a data vault modelled data warehouse. In order to ensure that the research results may be generalised to real-world situations, it was essential that the DBMS used in the experiment be representative of what is being used in practice.

The annual Magic Quadrant for Data Warehouse Database Management Systems report by Gartner Inc. is a good indicator of the major database management systems being used in the business intelligence and data warehousing sector. The following database management systems emerged from this report as candidates for the experiments (Feinberg & Beyer, 2011):

1. Oracle
2. IBM
3. Microsoft
4. Teradata

## 5. Sybase.

The above list is appears in order of market share according to Gartner Inc. (Petty, 2007). However, in view of the fact that the researcher is proficient in the Microsoft DBMS, SQL Server, it was deemed to be the logical candidate for the purposes of this study. The version of SQL Server that was used in the experiment was SQL Server 2008 R2 Enterprise Edition Service Pack 1. Annexure P contains some information on the way in which the DBMS was configured.

### 4.3.2 Operating system

The choice of an appropriate operating system for the experiment was dictated by the choice of the database management system in this case. SQL Server runs on a Microsoft operating system only and, thus, this limited the operating system candidates to Windows Server 2008 R2 or Windows 7. In order to be able to generalise the results of the research study, Windows Server 2008 R2 Enterprise Edition Service Pack 1 was selected in preference to Windows 7.

### 4.3.3 Eliminating uncontrolled variables

In order to reduce the number of uncontrolled variables that may have had an impact on the experiments, all non-essential windows services were disabled. This reduced the likelihood that a service or program running in the background would unexpectedly request CPU or memory during the execution of an experiment, thereby affecting the results of the experiment. A list of the services that was disabled may be found in Annexure A.

## 4.4 The dataset

### 4.4.1 Dataset requirements

It was essential that the dataset that was used in the experiments comply with the following requirements:

- The dataset had to be at least 10GB in size. If the dataset had been too small, a large percentage of the time measurements would have been in the millisecond range. Although the experiments had been designed in such a way that the researcher had had control over most of the variables that may have impacted on the

measurements, there were some variables, for example the operation system, which could not be fully controlled. These variables may, in turn, have unpredictably consumed memory, CPU time, or disk I/O during the execution of the experiment and this may have had an impact on the measurements. When the measurements are in the millisecond range, even the slightest interference from uncontrolled variables may have a major impact on the results of an experiment.

- The data model of the dataset had to be similar to that of a typical OLTP system. OLTP systems generally store data in a normalised data model (Mundy, Thornthwaite & Kimball, 2011:34), and, thus, the source data should also be stored in a normalised schema. This, in turn, ensures that the results of an experiment are generalisable.
- The data needed to be representative of data typically found in practice. This requirement also ensures that the results of the experiment are generalisable.
- A practical requirement involves the need for the dataset to be available to the researcher for experimentation.

#### **4.4.2 Dataset candidates**

Several candidate datasets were evaluated for the purposes of this study. These included:

- The AdventureWorks2008 sample database that ships with Microsoft SQL Server 2008.
- Sample data from real organisations.
- Dummy data that could be generated either programmatically or through specialised applications.
- Data from one of the Transaction Processing Performance Council (TPC) benchmarks.

AdventureWorks2008 is the sample OLTP dataset that Microsoft distributes with SQL Server 2008. It is based on a fictitious company named the Adventure Works Cycles. The dataset is readily available and can be downloaded for free from the internet.

AdventureWorks2008 met three of the requirements listed in section 4.4.1 above, namely, the dataset is normalised, it is representative of the data typically found in practice, and it is available for free on the internet. However, the dataset is only 80MB in size and generating

additional volumes of data would have been a highly laborious task. Accordingly, AdventureWorks2008 was not deemed to a suitable candidate dataset.

Sample data from real organisations would also have met three of the dataset requirements, namely, it would have been normalised, it is real data and, therefore, representative of the data found in practice, and the data volumes in real organisations are generally greater than was required for this experiment. However, privacy and confidentiality considerations are two reasons why organisations are reluctant to provide data to researchers, especially if they would not directly benefit from the research. This, in turn, means that datasets from real organisations are an unsuitable source for an experimental dataset.

Dummy data can be generated either programmatically or through specialised data generation applications. These applications are very effective as regards generating large volumes of data. However, it is extremely difficult to generate data that is both realistic and in a normalised form. Accordingly, this method was also deemed not to be viable.

The last alternative was a TPC dataset. TPC is a non-profit organisation. Its aim is to define objective benchmarks for database management systems (DBMS) (Anon., 2011b). There are currently three TPC benchmarks that are active, namely:

- TPC-C
- TPC-E
- TPC-H.

Both TPC-C and TPC-E are benchmarks for OLTP systems and provide normalised datasets while TPC-H is a decision support benchmark. Although TPC-C is still applicable as a benchmark, TPC-E is recommended as an OLTP benchmarks by the TPC as it has “been designed to be broadly representative of modern OLTP systems” (Anon., 2011a). TPC provides a data generator for each of these benchmarks that generates sample datasets. TPC-E meets all the requirements for an experimental dataset and, in view of its suitability, the TPC-E dataset has also been used in several other research experiments (Curino, Jones, Zhang & Madden, 2010:57; Ali, Rauf & Javed, 2010).

The only concern with the TPC-E dataset is that the data is perfect and there are no data quality issues nor is there missing data. However, this is hardly ever the case in practice (Kimball & Caserta, 2004:18).

### 4.4.3 The TPC-E dataset

The TPC-E dataset simulates “the activity of a brokerage firm that must manage customer accounts, execute customer trade orders, and be responsible for the interactions of customers with financial markets” (Anon., 2010:43). Trading is carried out by brokers on behalf of customers through customer accounts. Each customer has one or more accounts. Accounts trade in securities that are issued by companies while each security is traded at a particular stock exchange. Companies belong to an industry which, in turn, belongs to a sector (Anon., 2010:46).

Each trade request is categorised as either a buy or a sell and these are broken down further into trade types. The various trade types are presented in table 4.1 below.

**Table 4.1 TPC-E trade types (Anon., 2010:47)**

| Trade Type  | Request Category | Request Type | Description                                                                                           |
|-------------|------------------|--------------|-------------------------------------------------------------------------------------------------------|
| Limit-Buy   | Buy              | Limit Order  | Trade request to buy only when the market price is at or below the specified limit price.             |
| Limit-Sell  | Sell             | Limit Order  | Trade request to sell only when the market price is at or above the specified limit price.            |
| Market-Buy  | Buy              | Market Order | Trade request to buy immediately at the current market price.                                         |
| Market-Sell | Sell             | Market Order | Trade request to sell immediately at the current market price.                                        |
| Stop-Loss   | Sell             | Limit Order  | Trade request to sell only when (or if) the market price drops to or below the specified limit price. |

The customer may either use cash to buy securities – known as a cash account – or credit obtained from the brokerage firm – known as a margin account.

The brokerage firm also has quarterly financial data on each of the companies and daily data on the performance of all securities which are traded through the brokerage firm.

The TPC-E dataset is generated through an application known as EGenLoader which is provided by TPC as part of the TPC-E benchmark. Full details on the EGenLoader application and the parameters that were used to create the dataset for the experiments are contained in Annexure B.

#### 4.4.4 TPC-E data model

There are 33 TPC-E tables, each of which can be grouped into one of four groups. These groups are summarised in table 4.2 below.

**Table 4.2 TPC-E table groupings (Anon., 2010:43)**

| Table Grouping | Description                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------------------|
| Customer       | Tables containing customer related information.                                                              |
| Broker         | Tables containing data related to the brokerage firm and its brokers.                                        |
| Market         | Tables containing data related to the exchanges, companies, and securities that create the financial market. |
| Other          | Tables containing generic information that is referenced by multiple tables.                                 |

Figure 4.1 below illustrates the ERD for TPC-E (Anon, 2010). For the purpose of clarity, the relationships between the STATUS\_TYPE table and related tables have been omitted from the diagram. A full-size copy of the ERD may be found in Annexure C.



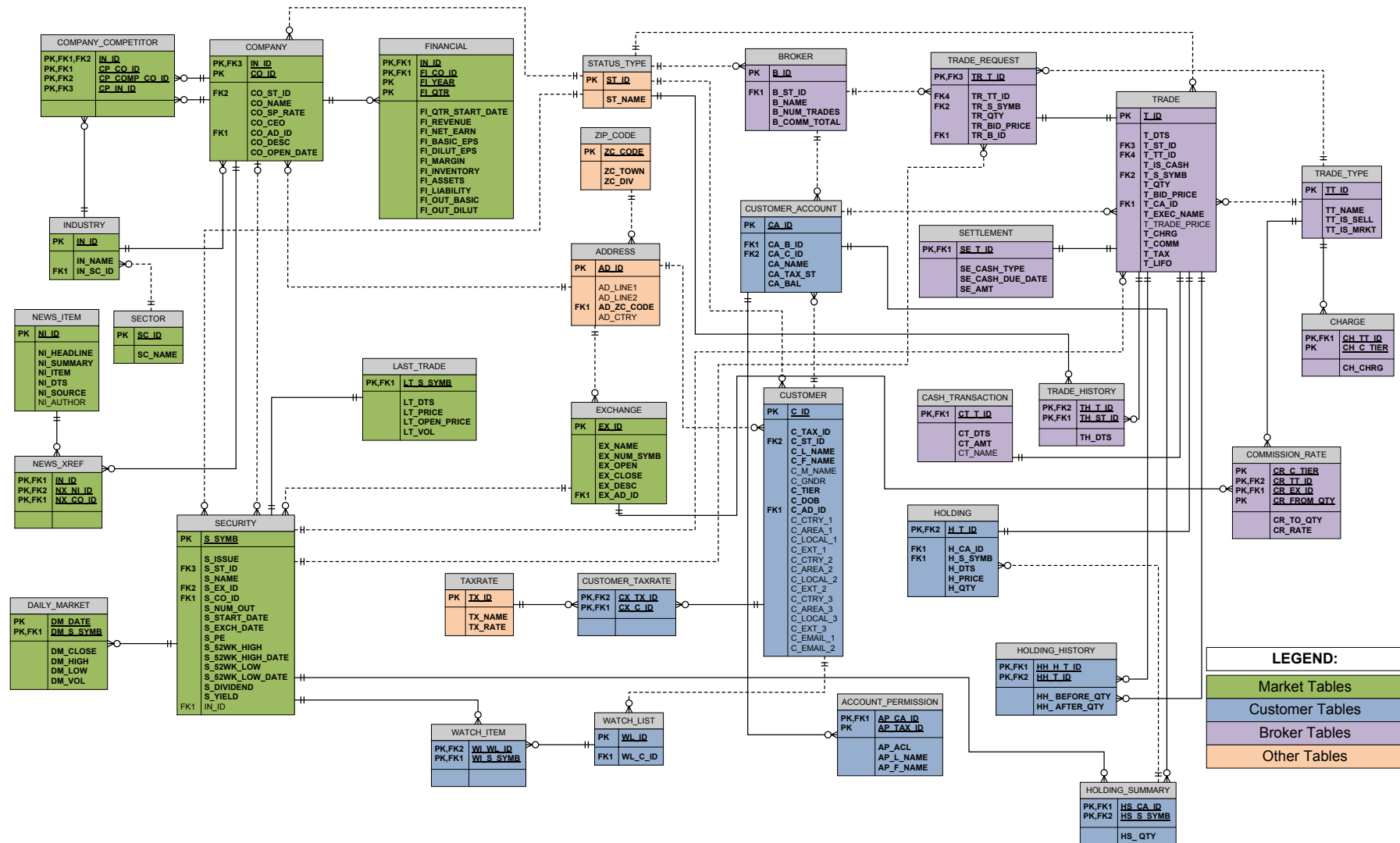


Figure 4.1 TPC-E entity relationship diagram

The description of the TPC-E tables is summarised in table 4.3 below. The data dictionary of all the TPC-E tables can be found in Annexure D.

**Table 4.3 Summary of subset of TPC-E tables**

| Table Name         | Description                                                                                                                                                                                                                               |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACCOUNT_PERMISSION | This table contains information about the access which the customer or an individual other than the customer has to a given customer account. Customer accounts may have trades executed on them by more than one person (Anon, 2010:51). |
| ADDRESS            | This table contains address information for companies, exchanges and customers. Each company, exchange and customer has one address record (Anon, 2010:63).                                                                               |
| BROKER             | The table contains information about brokers. There is one record for each broker (Anon, 2010:55).                                                                                                                                        |
| CASH_TRANSACTION   | The table contains information about cash transactions (Anon, 2010:55).                                                                                                                                                                   |
| CHARGE             | The table contains information about the charges for placing a trade request. Charges are based on the customer's tier and the trade type (Anon, 2010:55).                                                                                |
| COMMISSION_RATE    | The commission rate depends on several factors, including the tier the customer is in, the type of trade, the quantity of securities traded, and the exchange that executes the trade (Anon, 2010:56).                                    |
| COMPANY            | The table contains information about all companies with publicly traded securities. There is one record for each company (Anon, 2010:59).                                                                                                 |
| COMPANY_COMPETITOR | This table contains information for the competitors of a given company and the industry in which the company competes (Anon, 2010:59).                                                                                                    |
| CUSTOMER           | This table contains information about the customers of the brokerage firm. There is one row for each customer (Anon, 2010:52).                                                                                                            |
| CUSTOMER_ACCOUNT   | The table contains account information related to the accounts of each customer. Each customer has at least one account, but may have more than one account (Anon, 2010:52).                                                              |
| CUSTOMER_TAXRATE   | The table contains two references per customer into the TAXRATE table. One reference is for state/province tax; the other one is for national tax. The table contains the actual tax rates (Anon, 2010:53).                               |
| DAILY_MARKET       | The table contains daily market statistics for each security, using the closing market data from the last completed trading day. There is one record per security per day (Anon, 2010:60).                                                |
| EXCHANGE           | The table contains information about financial exchanges. There is one record for each exchange (Anon, 2010:60).                                                                                                                          |
| FINANCIAL          | The table contains information about a company's quarterly financial reports. There is one record per company per fiscal quarter (Anon, 2010:60).                                                                                         |
| HOLDING            | The table contains information about the customer account's security holdings (Anon, 2010:53).                                                                                                                                            |
| HOLDING_HISTORY    | The table contains information about holding positions that were inserted, updated or deleted and which trades made each change (Anon, 2010:53).                                                                                          |
| HOLDING_SUMMARY    | The table contains aggregate information about the customer account's security holdings (Anon, 2010:54).                                                                                                                                  |
| INDUSTRY           | The table contains information about the industries to which a company may belong. There is one record per industry. Each company belongs to one industry only (Anon, 2010:61).                                                           |

|               |                                                                                                                                                                                        |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LAST_TRADE    | The table contains one row for each security with the latest trade price and volume for each security (Anon, 2010:61).                                                                 |
| NEWS_ITEM     | The table contains information about news items of interest (Anon, 2010:62).                                                                                                           |
| NEWS_XREF     | The table contains a cross-reference of news items to the companies that are mentioned in the news item (Anon, 2010:62).                                                               |
| SECTOR        | The table contains information about market sectors that an industry belongs to. There is one record per sector. Each industry belongs to one sector only (Anon, 2010:62).             |
| SECURITY      | This table contains information about each security traded on any of the exchanges. A security is traded at one exchange only. There is one record for each security (Anon, 2010:63).  |
| SETTLEMENT    | The table contains information about how trades are settled: specifically whether the settlement is on margin or in cash and when the settlement is due (Anon, 2010:56).               |
| STATUS_TYPE   | This table contains status codes and descriptions for brokers, companies, customers, securities, and trade transactions. There is one record for each distinct status (Anon, 2010:64). |
| TAXRATE       | The table contains information about tax rates (Anon, 2010:64).                                                                                                                        |
| TRADE         | The table contains information about the trades that brokers carry out on behalf of customers. There is one record for every trade transaction that takes place (Anon, 2010:56).       |
| TRADE_HISTORY | The table contains the history of each trade transaction through the various states (Anon, 2010:57).                                                                                   |
| TRADE_REQUEST | The table contains information about pending limit trades that are waiting for a certain security price before the trades are submitted to the market (Anon, 2010:57).                 |
| TRADE_TYPE    | The table contains a list of valid trade types. There are two main categories of trades, namely, buy and sell (Anon, 2010:58).                                                         |
| WATCH_ITEM    | Each customer may create a list of securities that should be monitored. For every security that should be watched, one record is inserted into this table (Anon, 2010:54).             |
| WATCH_LIST    | Each customer may create a list of securities that should be monitored. A customer may have more than one list. For each list, one record is inserted into the table (Anon, 2010:64).  |
| ZIP_CODE      | The table contains zip and postal codes, towns, and divisions that accompany them (Anon, 2010:54).                                                                                     |

## 4.5 The user requirements

The dimensional model and data vault model that were designed for the experiments (also referred to as the target data models) were based on a set of fictitious requirements that had been created solely for the purpose of this experiment. In view of the fact that a user requirements specification was outside the scope of this study, a summary of the requirements only is provided in the section below. However, sufficient detail is included to enable the reader to validate the target data models against these requirements.

The first requirement stipulates that users should be able to analyse all the trade transactions. In other words, users should be able to slice and dice trade metrics by date, customer and account information, trade types, trade statuses, securities and the relevant

stock exchange. In addition, the metrics that users should be able to analyse include the quantity of securities traded, the bid and trade price of each transaction as well as the charge, commission and tax incurred on each transaction.

A second requirement is that users should be able to analyse the quarterly performance metrics of companies that list on the different exchanges. These performance metrics include earnings per basic and diluted shares, the average number of basic and diluted shares that are outstanding as well as revenue, net earnings, margin, inventory, assets and liabilities amounts. Users should be able to analyse these requirements by date and, more specifically, by fiscal quarters and by company.

Lastly, users need to look at the daily performance of securities that are traded on the different exchanges. Security performance metrics include the daily volume traded, daily high and low prices as well as the daily closing price. A high and low price over the preceding two weeks should also be included as well as the total volume traded over the preceding two weeks. These metrics need to be analysed by date, company, exchange and, finally, the security. Table 4.4 below summarises the attributes in respect of which changes in attribute values should be correctly reported over time.

**Table 4.4 Historical tracking requirement**

| Entity   | Attribute        |
|----------|------------------|
| Company  | CEO              |
| Company  | Credit rating    |
| Customer | Tier number      |
| Customer | Tier description |

The target data models that meet these requirements are provided in section 4.6 below.

## **4.6 Target data models**

This section describes the dimensional and data vault models that were used in the experiments.

#### **4.6.1 Dimensional model**

The Kimball methodology was used to create a dimensional model that met the set of requirements specified in section 4.5 above.

In order to ensure that the ERD for the dimensional model was easier to read, a separate ERD is shown for each fact table or star schema. It is important to note that the dimensions in these separate ERDs are conformed. A full size diagram of the complete model may be found in Annexure E.

The model was peer reviewed according to the user requirements by a well-known and experienced person in the data warehousing and business intelligence domain, Van Zyl Kruger. He found the model to be sound and in accordance with industry standards and best practice. A copy Kruger's report may be found in Annexure F.

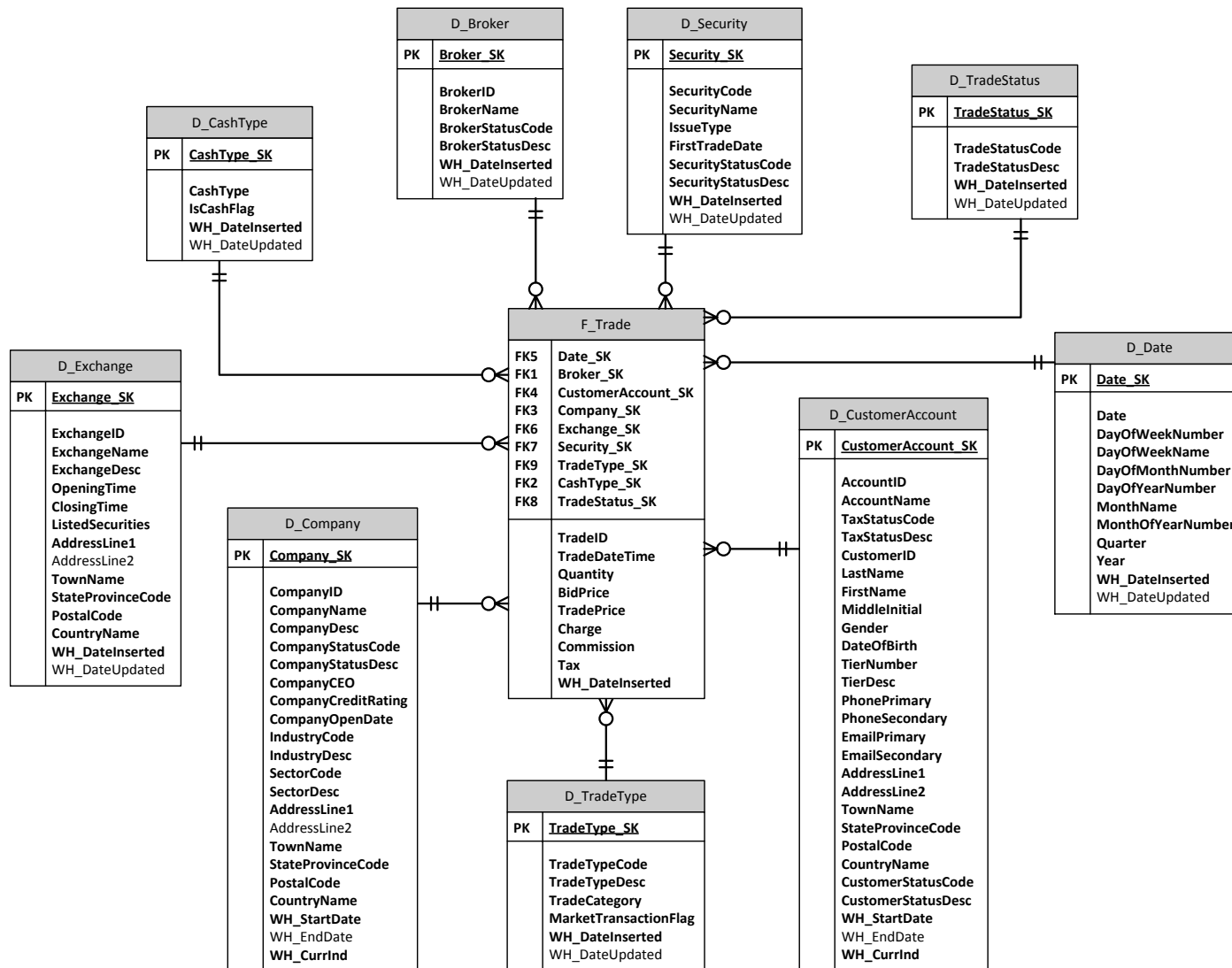


Figure 4.2 Dimensional model ERD - Trade fact table

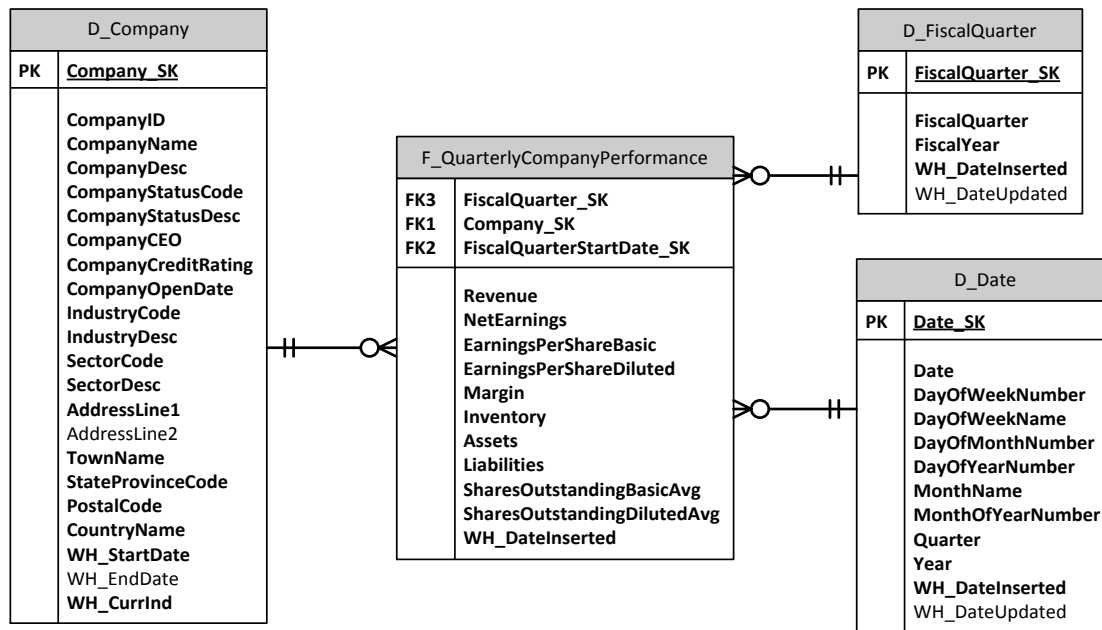


**Figure 4.3 Dimensional model ERD: company performance fact table**

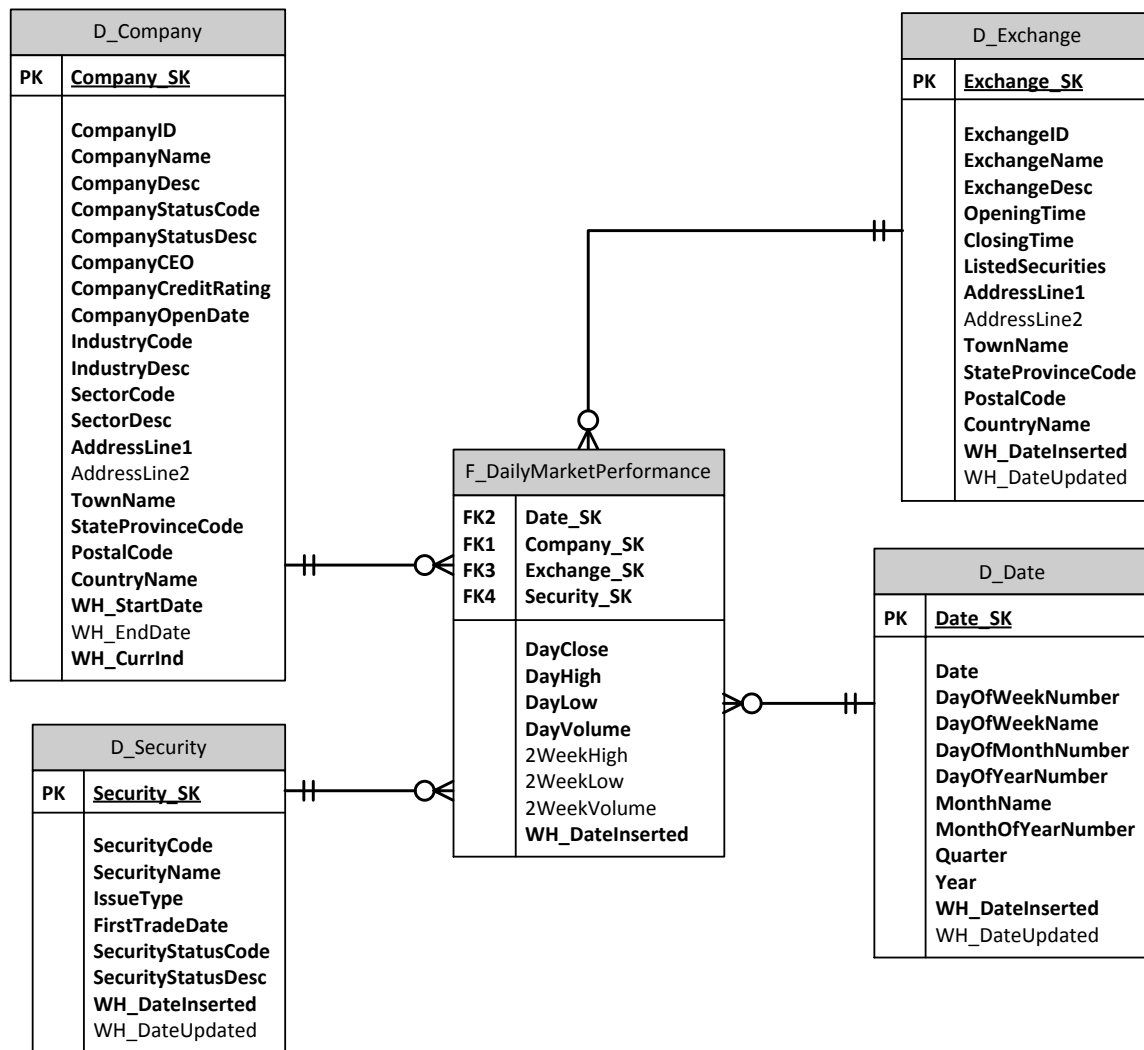


**Figure 4.4 Dimensional model ERD: market performance fact table**

The dimensional model was constructed following the four step dimensional modelling process as described by Kimball *et al.* (2008:246). The first data mart that was identified was trade. The grain of the trade fact table was defined such that each record in the trade fact table represented a single securities trade transaction at the lowest possible level. The dimensions and facts were identified according to the requirements. A similar process was followed for the other three data marts. The dimensions that the data marts had in common were all conformed.

Despite the fact that the logical primary key of each fact table consisted of a compound key made up of the entire collection of dimension foreign keys on the fact table (Kimball *et al.*, 2008:144), this is very rarely implemented. This is because the primary key is implemented as a unique clustered index in SQL Server by default. It is also possible to change the index



on the primary key fields to a non-clustered unique index. However, indexes consisting of all the dimension foreign keys would be expensive to maintain and would not yield optimal performance. Instead, a non-unique clustered index should be created on the date foreign key in the fact table. This, in turn, would minimise the fragmentation of data as new data would be inserted at the end of the fact table. Furthermore, single-column indexes on some of the most used dimension foreign keys may be created (Mundy *et al.*, 2011:149).

The indexing strategy as described above was implemented on all three fact tables. A clustered index was created on Date\_SK for F\_DailyMarketPerformance while single-column non-clustered indexes were created on Company\_SK and Security\_SK. For F\_QuarterlyCompanyPerformance, a clustered index was created on FiscalQuarter\_SK while single-column non-clustered indexes were created on Company\_SK and FiscalQuarterStartDate\_SK. With regards to F\_Trade, a clustered index was created on Date\_SK while single-column non-clustered indexes were created on Broker\_SK, CustomerAccount\_SK, Company\_SK and Security\_SK. Note that no indexes were created on either Exchange\_SK, CashType\_SK, TradeType\_SK or TradeStatus\_SK fields. These are low cardinality tables and, according to Mundy *et al.* (2011:149), indexes on low cardinality fields add little value.

Each table in the dimensional model has some standard auditing fields that are prefixed with “WH\_”. These are summarised in table 4.5 below:

**Table 4.5 Dimensional model system fields summary**

| Field           | Data Type    | Allow Nulls | Purpose                                                                                                                                                                       |
|-----------------|--------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WH_DateInserted | DATETIME2(0) | No          | Timestamp of when the record was inserted into the table. Applicable to all tables except type 2 SCDs (where its function is taken over by WH_StartDate).                     |
| WH_DateUpdated  | DATETIME2(0) | Yes         | Timestamp of when the record was last updated. Not applicable to type 2 SCDs (where its function is taken over by WH_StartDate) or fact tables where records are not updated. |
| WH_StartDate    | DATETIME2(0) | No          | The effective start date of the record. Applicable to type 2 SCDs only.                                                                                                       |
| WH_EndDate      | DATETIME2(0) | Yes         | The effective end date of the record. Applicable to type 2 SCDs only.                                                                                                         |
| WH_CurrInd      | BIT          | No          | Flags current records. Applicable to type 2 SCDs only.                                                                                                                        |

The source-to-target map, as described by Kimball *et al.* (2008:306), may be found in Annexure G.

#### **4.6.2 Data vault model**

The data vault model was created according to the methodology set out by Linstedt and Graziano (2011) in order to meet the same set of requirements that was used for the dimensional model. Care was taken to ensure that both the models met the requirements exactly so that they were comparable.

The ERD of the data vault model is depicted in figure 4.5 below. A full size diagram of the model may be found in Annexure H. The model was reviewed by one of the leading academics and practitioners of the data vault, Martijn Evers. Evers found the data model to comply with the standards of data vault modelling as set out by Linstedt and Graziano (2011). A copy of Evers' report may be found in Annexure K.

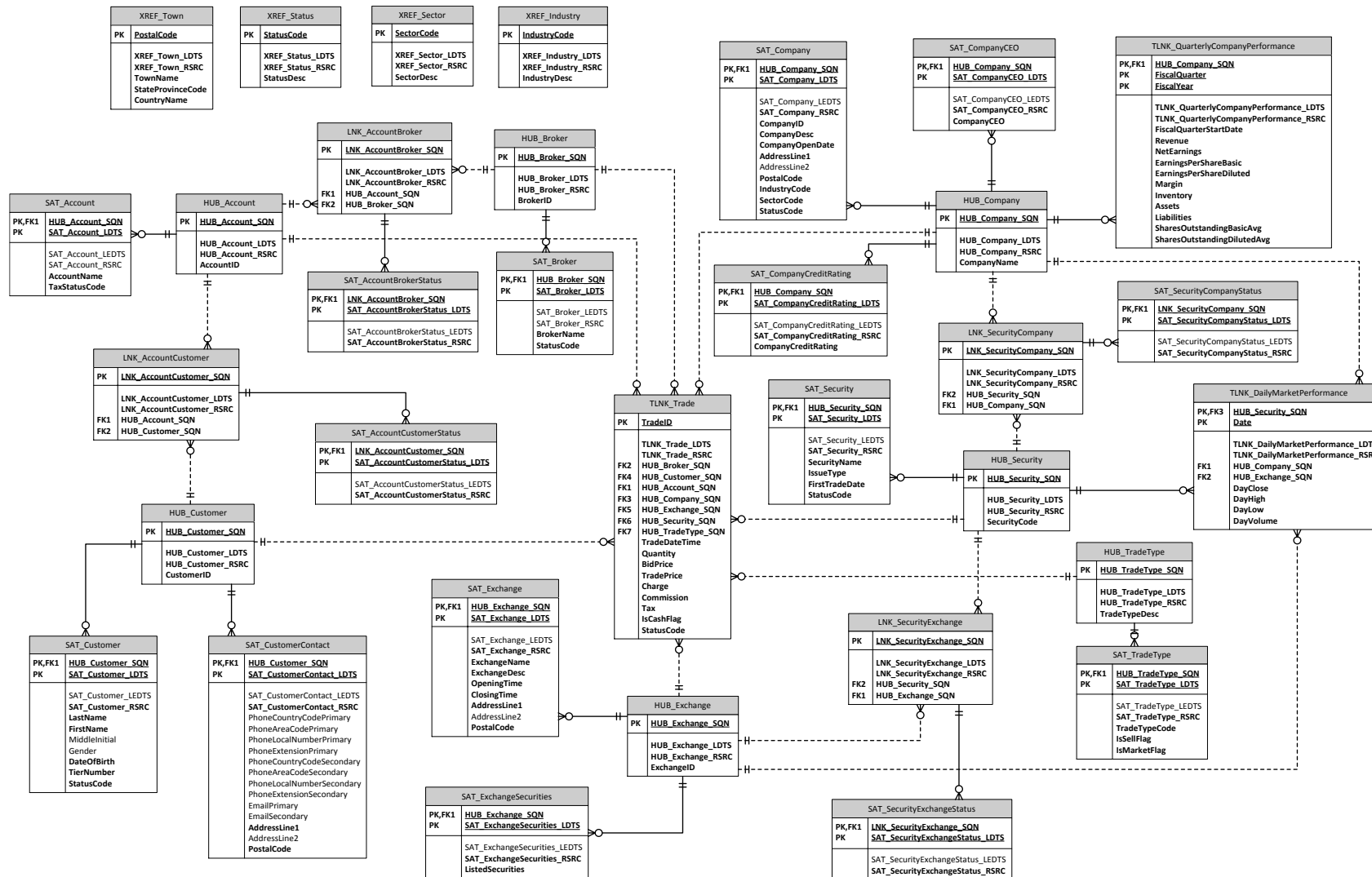


Figure 4.5 Data vault model ERD

The seven-step method, as described in chapter 3 section 3.6.7, was followed in creating the data vault model. The hubs that were identified included account, broker, company, customer, exchange, security and trade type. The business keys for all the hubs were the source system primary key with the exception of company and trade type. The source system primary keys were selected because they were being used by the business to uniquely identify instances of the entity. For company and trade type, the business used a description rather than an ID field.

In the second step, the relationships between the hub entities were identified. This resulted in the modelling of the links as listed in table 4.6 below.

**Table 4.6 Data vault model resulting link entities**

| Link Entity                      | Relationship                                                          |
|----------------------------------|-----------------------------------------------------------------------|
| LNK_AccountBroker                | Account and broker                                                    |
| LNK_AccountCustomer              | Account and customer                                                  |
| LNK_SecurityCompany              | Security and company                                                  |
| LNK_SecurityExchange             | Security and exchange                                                 |
| TLNK_DailyMarketPerformance      | Security, company and exchange                                        |
| TLNK_QuarterlyCompanyPerformance | Company                                                               |
| TLNK_Trade                       | Broker, customer, account, company, exchange, security and trade type |

The third step involved the modelling of satellite entities for both hub and link entities. The resulting satellite entities are listed in table 4.7 below.

**Table 4.7 Data vault model resulting satellite entities**

| Parent Entity                    | Satellite Entity                |
|----------------------------------|---------------------------------|
| HUB_Account                      | SAT_Account                     |
| HUB_Broker                       | SAT_Broker                      |
| HUB_Company                      | SAT_Company                     |
| HUB_Customer                     | SAT_Customer                    |
| HUB_Exchange                     | SAT_Exchange                    |
| HUB_Security                     | SAT_Security                    |
| HUB_TradeType                    | SAT_TradeType                   |
| LNK_AccountBroker                | SAT_AccountBrokerStatus         |
| LNK_AccountCustomer              | SAT_AccountCustomerStatus       |
| LNK_SecurityCompany              | SAT_SecurityCompanyStatus       |
| LNK_SecurityExchange             | SAT_SecurityExchangeStatus      |
| TLNK_DailyMarketPerformance      | SAT_DailyMarketPerformance      |
| TLNK_QuarterlyCompanyPerformance | SAT_QuarterlyCompanyPerformance |
| TLNK_Trade                       | SAT_Trade                       |

The fourth step involved the normalisation of satellites based on either source system, expected frequency of change, or a combination of both. Since the dataset in the experiment came from a single source system, the satellites were normalised based on the expected frequency of change. As a result, SAT\_Company was divided into three satellite tables, namely, SAT\_Company, SAT\_CompanyCEO and SAT\_CompanyCreditRating. This was as a result of the fact that the company name, location, date it was opened, etc. were expected to be fairly stable. However, a company will generally have multiple CEOs over its lifespan while credit ratings may change annually. On a similar basis, SAT\_Customer was split into SAT\_Customer and SAT\_CustomerContact while HUB\_Exchange was split into HUB\_Exchange and HUB\_ExchangeSecurities.

Step five was skipped as no link-to-link relationships were identified. Step six involved the transformation of transactional links to either type 1 or type 2 transactional links. Recall the difference between type 1 and type 2 transactional links as detailed in chapter 3, section 3.6.3. Type 1 transactional link entities remove the load end date timestamp from the related satellite, and also remove the load date timestamp from the satellite entity's primary key. This, in turn, changes the relationship between the link entity and satellite entity from on-to-many to one-to-one, thus enabling the satellite to store one record only for each record in the link entity. Type 2 transactional link entities denormalise the model and collapse the information of the satellite entity into the transactional link entity. The researcher assumed that the data destined for both the transactional link entities, as well as its related satellite entities, would arrive in the data vault at the same time. Accordingly, the decision was made to transform the transactional link

entities into type 2 transactional link entities and, thus, SAT\_DailyMarketPerformance, SAT\_QuarterlyCompanyPerformance and SAT\_Trade were collapsed into TLNK\_DailyMarketPerformance, TLNK\_QuarterlyCompanyPerformance and TLNK\_Trade respectively.

The last step in the seven-step method requires cross-reference tables to be added. The cross-reference tables that were added were XREF\_Industry, XREF\_Sector, XREF\_Status and XREF\_Town. These were modelled as cross-reference tables because they were code description tables that were used frequently throughout the model and they, thus, complied with Linstedt's definition of cross-reference tables (Linstedt & Graziano, 2011:146).

The indexing strategy that was adopted in the implementation of the data vault target data model was in accordance with the recommendation of Evers (2011). Unique clustered indexes were built on hub entity sequence numbers while unique non-clustered indexes were built on the hub business key fields. Unique clustered indexes were built on link sequence numbers and non-unique non-clustered indexes on each of the hub entity foreign keys in the link entity. Composite unique clustered indexes were built on all satellite entities comprising the hub entity foreign key followed by the load date time stamp. For satellites that referenced the cross-reference tables, a non-clustered index was created on the cross reference code field in the satellite entity. On each cross-reference table, a clustered unique index on the code field, which was also the primary key, was built. There were some exceptions to Evers' (2011) recommended initial indexing strategy for data vault models. Indexes on low cardinality fields were omitted, as recommended by Mundy *et al.* (2011:149). The non-unique non-clustered indexes on the HUB\_Exchange\_SQN foreign key were omitted in all link entities with foreign key relationships to HUB\_Exchange, as there are only four distinct values in the table. The exclusion included LNK\_SecurityExchange, TLNK\_DailyMarketPerformance and TLNK\_Trade. Similarly, the non-unique non-clustered index on HUB\_TradeType\_SQN in TLNK\_Trade was also omitted.

## 4.7 Experiments

The following section contains details on the four experiments that were conducted. For each experiment the research question is defined, the hypothesis is formulated and the results are predicted. The execution of the experiment is described and the statistical analysis of the results is discussed.

#### 4.7.1 Experimental design

Each of the experiments below was based on a randomised, post-test only, control group design. Conceptually, a treatment or intervention was applied to the dimensional model. This treatment involved converting the dimensional model to a data vault model. The control group remained the unchanged dimensional model as treatments are never applied to control groups (Trochim, 2012). This resulted in two groups, a dimensional model group and a data vault model group, which enabled the researcher to compare the observations or measurements arising from the two models.

Random assignment assigns testing units randomly to a predefined number of groups. As a result there is a high probability that the groups are equal. In the experiments described below, examples of testing units include the historical data set, each of the incremental datasets, each of the queries, and so forth. As mentioned in chapter 2, section 2.5.1, despite the fact that the testing units were not randomly assigned to either of the groups they were still equal. This is because the testing units were either exactly the same, as is the case of the historical or incremental datasets, or they were equal in that the results they returned were exactly the same, as is the case of the queries. It is for this reason that the experimental design is known as a randomised design.

The only difference between the experimental designs for each of the experiments was that the number of observations was different. As an example, consider the experimental design for the load performance experiment, as depicted in figure 4.6 below. Ten observations are depicted because the historical dataset was loaded 10 times and measured after each load. The incremental load performance, depicted in figure 4.6 below, has 90 observations as there were 30 incremental datasets, each loaded three times. It is for this reason that the experimental designs will not be depicted graphically for each experiment.

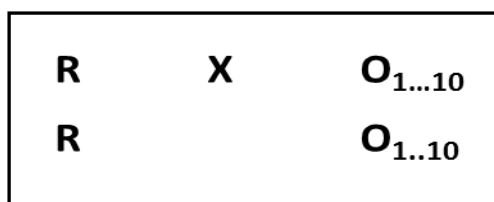


**Figure 4.6 History load experimental design**

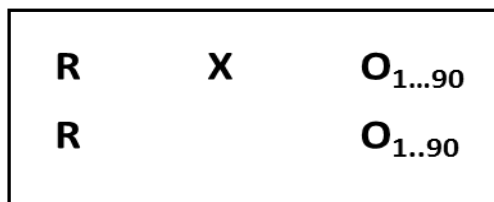


**Figure 4.7 Incremental load experimental design**

#### **4.7.2 Statistical analysis methods**

Two statistical analysis methods were used to analyse the data which resulted from the experiments, namely, the t-test for dependent samples and the Wilcoxon matched pairs test.

The t-test determines whether “the means of two groups are statistically different from each other” (Trochim, 2012). The t-test for dependent samples is used when specific observations/measurements from one group are related to observations/measurements from another group. As an example of dependent samples, consider the query experiment in which five distinct query categories were defined. From the total set of observations across all categories, observations from category 1 of the dimensional model queries were related to observations from category 1 from the data vault queries. The t-test assumes that the observations are normally distributed (Steyn, Smit, Du Toit & Strasheim, 1998:429). In cases in which it is either not possible to assume a normal distribution or in which the number of observations is small, the Wilcoxon matched pairs test is an acceptable alternative. The test is very similar to the t-test but instead of comparing the means of two groups, it compares the mean ranks (Steyn *et al.*, 1998:598).

#### **4.7.3 Load performance**

The load performance experiment was the first experiment that was conducted.

##### **4.7.3.1 Define the research question**

The research question that this particular experiment seeks to answer is how the two target models compare in terms of load performance, that is, loading data from the source system into the two different data models.

There are two distinct scenarios in terms of which data is loaded into a data warehouse, namely, the historical and the incremental loads (Kimball *et al.*, 2008:437). Historical loads are a once-off occurrence that loads the initial dataset into the data warehouse while, on the other



hand, incremental loads happen each time that the data in the warehouse is refreshed. This usually occurs on a daily basis but, in near real-time systems incremental loads may occur every couple of minutes. Accordingly, the impact of incremental load performance on a data warehouse is more than that of a once off historical load. Thus, in view of these distinctions, the research question for this experiment seeks to compare historical and incremental load performance independently for each of the two target data models (dimensional model and data vault model).

#### **4.7.3.2 Formulate the hypotheses**

The hypotheses for this experiment was formulated as follow:

- The mean time required to load historical data into a data vault model is less than the mean time required to load the same data into a dimensional model.
- The mean time required to load incremental data into a data vault model is less than the mean time required to load the same data into a dimensional model.

Of the two hypotheses, the second hypothesis should have a bigger impact on the result of the experiment than the first hypothesis as it affects the day-to-day lifecycle of a data warehouse.

#### **4.7.3.3 Predict the results**

The researcher took the following aspects into account when predicting the outcome of the experiment:

- In many instances, the data vault model incorporates several smaller tables that may be loaded in parallel. This parallelisation is at the heart of the prediction. As an example, compare the data vault tables that contain the same information as D\_Company. These tables are HUB\_Company, SAT\_Company, SAT\_CompanyCEO, SAT\_CompanyCreditRating, XREF\_Industry, XREF\_Sector, XREF\_Status and XREF\_Town. The hub entity and the four cross-reference entities may be loaded independently and in parallel. Thereafter, the two satellite entities may be loaded independently and in parallel.
- The data vault model does not have as much redundant data to load as the dimensional model. In the previous example, the four cross-reference tables are loaded only once. In the resulting dimensional model, the values are loaded for several tables, including D\_Broker, D\_Company, D\_CustomerAccount, D\_Exchange, D\_Security and D\_TradeStatus. As an example, consider the StatusDesc attribute in XREF\_Status. There is only one record in this table where the StatusDesc equals "Active". Compare

this to the BrokerStatus attribute in the D\_Broker dimensions with over 500 broker records, most of which are active. The “Active” description is repeated for every active broker record. Data redundancy in the dimensional model is a side-effect of denormalisation.

- Another impact of less redundancy in the case of the data vault when compared to the dimensional model is that there is less data that needs to be written to disk. The bottleneck which occurs in most data warehouse implementations is disk I/O. Thus, the less data there is to be written to disk, the less the negative impact that disk I/O will have on the loading of data.
- It also needs to be taken into account that less data transformations are required when loading data from the source system into the data vault model as compared to the dimensional model. This is as a result of the fact that business rules are not implemented when loading data into the data vault, although they are implemented when querying from the data vault. This becomes evident if the source-to-target mapping for both the dimensional model and the data vault model are compared. The source-to-target mappings for both models may be found in Annexures G and J respectively. As an example, consider the following t-SQL extract that populates the CountryName field on D\_Company. These business rules are not implemented when loading data into the data vault, but rather when data is queried from the data vault.

```
CASE UPPER (a.AD_CTRY)
      WHEN 'USA' THEN UPPER (a.AD_CTRY)
      WHEN NULL THEN 'Unknown'
      ELSE
        UPPER (LEFT (a.AD_CTRY,1)) +
        LOWER (RIGHT (a.AD_CTRY, LEN (a.AD_CTRY) -1))
END AS CountryName
```

- The last point on which the prediction is based is the fact that the date and fiscal quarter dimensions (D\_Date and D\_FiscalQuarter) in the dimensional model are not required in the data vault model. In view of the fact that these dimensions are loaded on a once-off basis only, it will play a role in the historical loads only.

Based on the above considerations, the researcher predicted that the mean time required to load both historical and incremental data into a data vault model would be less than the mean time required to load the same data into a dimensional model.

#### 4.7.3.4 Execute the experiment

This section describes the way in which the experiment was designed and executed. The technical detail in respect of the exact implementation of the experiment is described in the experiment protocol in Annexure L. The source system SQL Server database, known as TPCE (figure 4.1), was initially populated with history data for 10 days. The two target data models were implemented in two separate SQL Server databases known as TPCE\_DM (figure 4.2) and TPCE\_DV (figure 4.5).

For the history load performance experiment, all the data from the source system was extracted and loaded into both the target data models. This was done by writing simple t-SQL in the form of INSERT INTO...SELECT FROM statements. Date time stamps were recorded both before and after query execution. In addition, the duration of each query was calculated. This process was repeated 10 times, truncating all tables prior to each load to ensure that the results were reliable. All queries were executed sequentially while the SQL Server service was restarted before the execution of each query. This was to ensure that SQL Server's caching mechanisms did not interfere with the results.

The tables were not truncated after the final iteration of the history load. The incremental source data was loaded into the TPCE source database for one additional day and the data from that day was then incrementally loaded into each of the target data models. This was done through t-SQL statements that made use of a combination of database views, INSERT INTO...SELECT FROM statements and MERGE INTO statements. All the transformations and joins from the TPCE database were implemented through the database views. The MERGE statement was then used to update dimensions in the dimensional model and hub, link (excluding transactional link) and satellite entities in the data vault model. INSERT INTO...SELECT FROM statements were used to load the fact tables and the transactional link entities incrementally into the dimensional and data vault models respectively. All non-clustered indexes were disabled on the fact and transactional link tables before data was inserted and rebuilt after the data had been loaded. Date time stamps were recorded both before and after query execution. In addition, the duration of each query was calculated. Incremental loads for 30 sequential days were executed. The process was repeated three times, reverting back to the history dataset only at the beginning of each iteration. Repetition was implemented in order to ensure that the results were reliable. All queries were executed sequentially and the SQL Server service was restarted before the execution of each query. This was to ensure that SQL Server's caching mechanisms did not interfere with the results.

#### 4.7.3.5 Results and analysis

A sample of the raw result set of the history load is presented in table 4.8 below. Although each of these result rows was obtained through the sequential execution of each of the history load queries, the data was prepared for analysis to reflect the fact that some queries are typically executed in parallel. As an example, consider a dimensionally modelled data warehouse where dimensions are loaded in parallel and fact tables are often loaded sequentially. Similarly, for the data vault model hub and cross reference entities are loaded in parallel, followed by the link and satellite entities and, lastly, the transactional link entities. In order to reflect this accurately, the total duration for the dimensional model history and incremental load was calculated as  $Total\ DM\ Duration = Max(DimDuration) + \sum FactDuration$ . The total duration for the data vault model history and incremental load was calculated as  $TotalDVDuration = Max(HubAndXREFDuration) + Max(SatDuration) + \sum TLNKDuration$ .

**Table 4.8 Sample raw result set for history data load experiment**

| Iteration | Model | TableName                   | TableType | DurationSeconds |
|-----------|-------|-----------------------------|-----------|-----------------|
| 1         | DM    | D_Broker                    | Dimension | 0,015           |
| 1         | DM    | D_CashType                  | Dimension | 4,56            |
| 1         | DM    | D_Company                   | Dimension | 0,742           |
| 1         | DM    | D_CustomerAccount           | Dimension | 6,644           |
| 1         | DM    | D_Date                      | Dimension | 0,659           |
| 1         | DM    | D_Exchange                  | Dimension | 0,011           |
| 1         | DM    | D_FiscalQuarter             | Dimension | 0,003           |
| 1         | DM    | D_Security                  | Dimension | 0,449           |
| 1         | DM    | D_TradeStatus               | Dimension | 0,003           |
| 1         | DM    | D_TradeType                 | Dimension | 0,014           |
| 1         | DM    | F_DailyMarketPerformance    | Fact      | 698,099         |
| 1         | DM    | F_QuarterCompanyPerformance | Fact      | 4,059           |
| 1         | DM    | F_Trade                     | Fact      | 139,264         |

A summary of the history load and incremental load results are presented in table 4.9 and Table 4.10 below respectively.

**Table 4.9 Historical load results in seconds**

| Iteration | Duration_DM | Duration_DV |
|-----------|-------------|-------------|
| 1         | 848.066     | 224.453     |
| 2         | 852.858     | 225.077     |
| 3         | 855.913     | 224.742     |
| 4         | 846.592     | 225.881     |
| 5         | 847.086     | 224.022     |
| 6         | 847.998     | 223.179     |
| 7         | 848.683     | 224.207     |
| 8         | 867.014     | 221.700     |
| 9         | 851.310     | 227.745     |
| 10        | 849.826     | 225.398     |

**Table 4.10 Incremental load results in seconds**

| Day | Duration_DM | Duration_DV |
|-----|-------------|-------------|
| 1   | 71.326      | 54.850      |
| 2   | 72.921      | 56.925      |
| 3   | 78.143      | 60.087      |
| 4   | 79.594      | 62.632      |
| 5   | 87.768      | 66.652      |
| 6   | 90.627      | 71.671      |
| 7   | 93.522      | 74.376      |
| 8   | 96.193      | 76.347      |
| 9   | 97.331      | 79.087      |
| 10  | 103.362     | 82.006      |
| 11  | 107.983     | 88.890      |
| 12  | 114.745     | 90.711      |
| 13  | 120.249     | 96.249      |
| 14  | 121.939     | 102.231     |
| 15  | 119.634     | 102.304     |
| 16  | 123.830     | 104.618     |
| 17  | 135.529     | 108.784     |
| 18  | 138.882     | 114.945     |
| 19  | 145.643     | 131.188     |
| 20  | 140.372     | 133.465     |
| 21  | 156.898     | 152.029     |
| 22  | 173.549     | 173.885     |
| 23  | 203.989     | 173.446     |
| 24  | 223.890     | 206.371     |
| 25  | 283.695     | 254.364     |
| 26  | 316.566     | 303.076     |
| 27  | 366.749     | 355.784     |
| 28  | 401.016     | 412.401     |
| 29  | 427.646     | 430.899     |
| 30  | 458.346     | 433.886     |

At first glance it would appear that the results indicate that the data vault model outperformed the dimensional model with regard to historical and incremental data loads. Table 4.11 below presents the results of a Wilcoxon matched pairs test for the history load results from Statistica. A p-value of 0.005062 indicates that the differences in values between the history load results for the dimensional and data vault models are statistically significant. This was confirmed by the results of a t-test for dependent samples, as presented in table 4.12 below and also depicted in the box and whisker plot in figure 4.8 below. This, in turn, supports the hypothesis that the mean time required to load historical data into a data vault model is less than the mean time required to load the same data into a dimensional model.

**Table 4.11 History load results – Wilcoxon matched pairs test**

| Pairs of Variables        | Wilcoxon Matched Pairs Test (History Load Results)<br>Marked tests are significant at p < .05000 |      |          |          |
|---------------------------|--------------------------------------------------------------------------------------------------|------|----------|----------|
|                           | Valid N                                                                                          | T    | Z        | p-value  |
| Duration_DM & Duration_DV | 10                                                                                               | 0.00 | 2.803060 | 0.005062 |

**Table 4.12 History load results – T-test for dependent samples**

| Variable    | t-test for Dependent Samples (History Load Results)<br>Marked differences are significant at p < .05000 |         |    |         |               |         |       |       |                    |                     |
|-------------|---------------------------------------------------------------------------------------------------------|---------|----|---------|---------------|---------|-------|-------|--------------------|---------------------|
|             | Mean                                                                                                    | Std.Dv. | N  | Diff.   | Std.Dv. Diff. | t       | df    | p     | Confidence 95.000% | Confidence +95.000% |
| Duration_DM | 851.535                                                                                                 | 6.150   |    |         |               |         |       |       |                    |                     |
| Duration_DV | 224.640                                                                                                 | 1.610   | 10 | 626.894 | 7.066         | 280.557 | 9.000 | 0.000 | 621.840            | 631.949             |

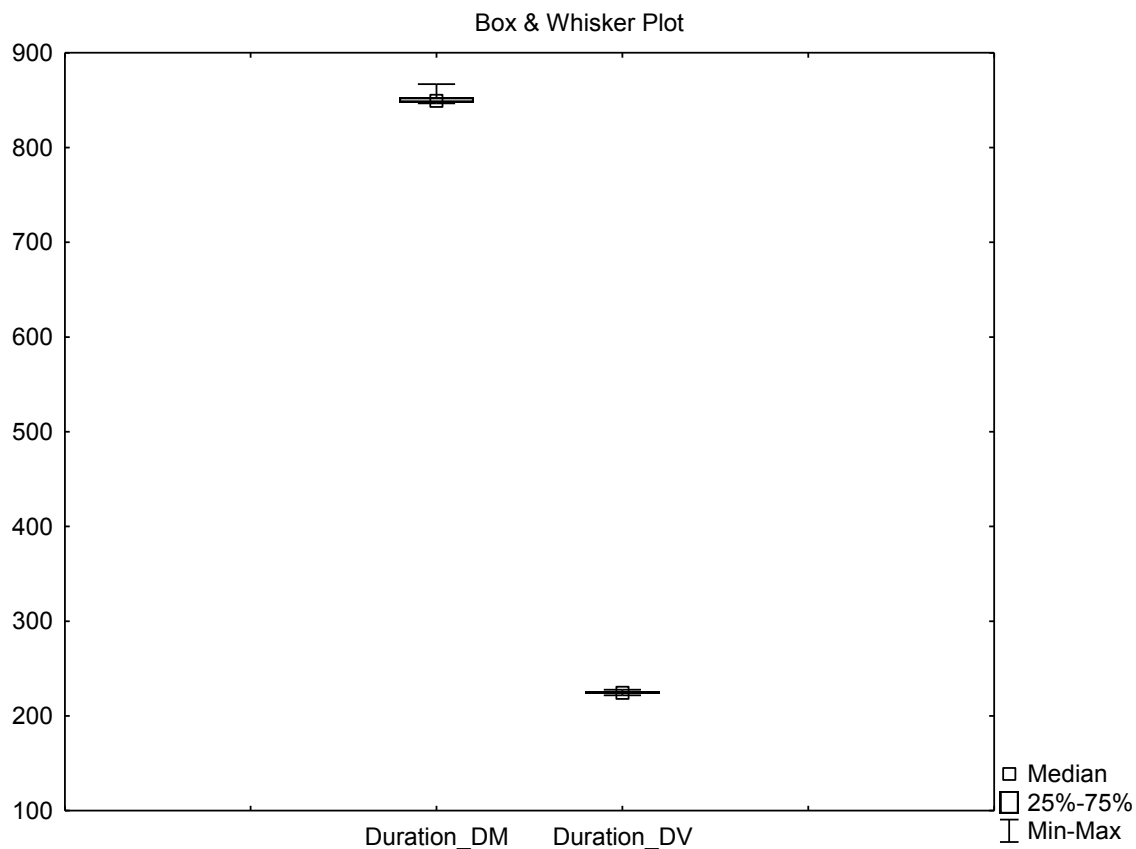


**Figure 4.8 History load results: box-and-whisker plot**

Table 4.13 below presents the results of a t-test for dependent samples for the incremental load results from Statistica. A p-value of 0.000 indicates that the differences in duration between the dimensional and data vault models are statistically significant for incremental loads. This is

visually confirmed by the box and whisker plot depicted in figure 4.9 below. This, in turn, supports the hypothesis that the mean time required to load incremental data into a data vault model is less than the mean time required to load the same data into a dimensional model.

**Table 4.13 Incremental load results: t-test for dependent samples**

| Variable    | t-test for Dependent Samples (Query Results)<br>Marked differences are significant at $p < .05000$ |         |    |        |               |        |    |          |                     |                     |
|-------------|----------------------------------------------------------------------------------------------------|---------|----|--------|---------------|--------|----|----------|---------------------|---------------------|
|             | Mean                                                                                               | Std.Dv. | N  | Diff.  | Std.Dv. Diff. | t      | df | p        | Confidence -95.000% | Confidence +95.000% |
| Duration_DM | 171.731                                                                                            | 112.536 |    |        |               |        |    |          |                     |                     |
| Duration_DV | 155.139                                                                                            | 116.516 | 90 | 16.593 | 12.072        | 13.039 | 89 | 0.000000 | 14.064              | 19.121              |

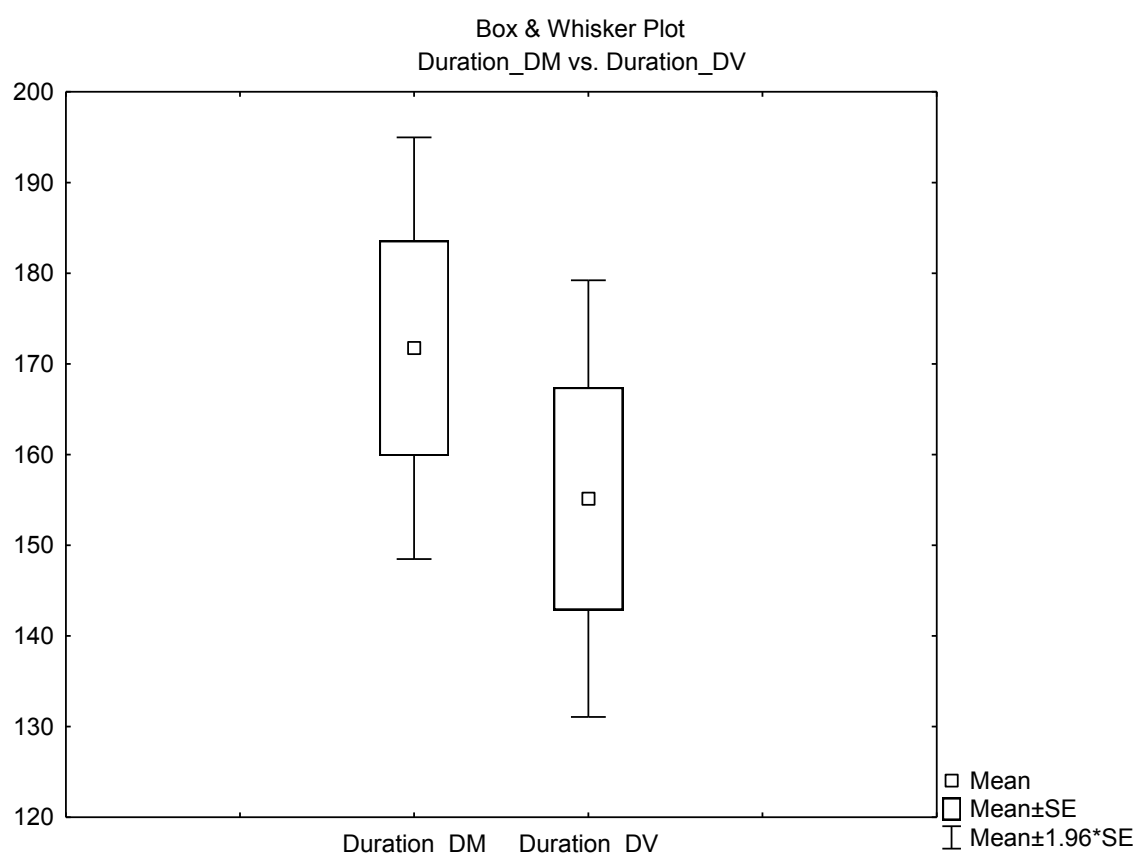


**Figure 4.9 Incremental load results: box-and-whisker plot**

#### 4.7.3.6 Conclusion

The results of the experiment support the hypothesis that the mean time required to load data into a data vault model is less than the mean time required to load the same data into a dimensional model. This is true for both historical and incremental loads. Accordingly, both the



hypotheses stated in section 4.7.3.2 above are accepted. It is significant that the data vault model performed markedly better than the dimensional model for historical loads relative to incremental loads.

#### 4.7.4 Query performance

The query performance experiment was the next experiment that was conducted.

##### 4.7.4.1 Define the research question

The research question that this particular experiment seeks to answer is how the two target models compare in terms of query performance.

##### 4.7.4.2 Formulate the hypothesis

The hypothesis for this experiment may be formulated as follow:

*The mean query execution time for queries to a data vault model is less than the mean query execution time for similar queries to the dimensional model.*

Queries to a data warehouse may differ in complexity. For example, a query to a single dimension table will be less costly than a query that joins two large fact tables through their conformed dimensions. Accordingly, it would not make sense to compare these two queries in terms of the query execution time and for this reason, the researcher defined different types of queries in terms of their complexity. Table 4.14 below lists the categories that were used for the purposes of this experiment and describes the categories in terms of a dimensional model. It should be noted that these categories were identified solely for the purposes of this experiment and are not generally accepted query categories.

**Table 4.14 Query categories in terms of dimensional models**

| Category | Description                                                                                                                                                          | # Queries |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| 1        | Simple SELECT statement to a single dimension table. The only field allowed in a WHERE clause is a current indicator flag.                                           | 3         |
| 2        | Aggregated queries to a single dimension table that include WHERE, GROUP BY, HAVING and ORDER by clauses.                                                            | 3         |
| 3        | Queries the relationship between two dimension tables through a single fact table. Queries include SELECT DISTINCT, WHERE and ORDER BY clauses.                      | 1         |
| 4        | Non-aggregated queries to a single fact table and all its related dimension tables. Queries include WHERE and ORDER BY clauses.                                      | 2         |
| 5        | Aggregated queries to a single fact table and some of its related dimension tables. Queries include aggregation functions, calculations, WHERE and GROUP BY clauses. | 2         |

The hypothesis will be tested for each of the different categories independently and also for all the categories combined.

#### 4.7.4.3 Predict the results

The researcher expected that the data vault model would either match the dimensional model or even perform marginally better than the dimensional model for the lower category queries, that is, the simpler queries. However, this would be purely a function of the number of tables that were queried. As an example, consider a query that retrieves a distinct list of customer numbers from the data warehouse. Sample queries for the dimensional as well as the data vault model are presented in table 4.15 below. In this particular example, the data vault query was expected to execute faster than the dimensional model query as it is a smaller table (it does not go down to customer account level) and it contains no WHERE clause.

**Table 4.15 Category 1 query comparison**

|               | Dimensional Model                                                                        | Data Vault Model                                           |
|---------------|------------------------------------------------------------------------------------------|------------------------------------------------------------|
| Query         | <pre>SELECT     CustomerID FROM     dbo.D_CustomerAccount WHERE     WH_CurrInd = 1</pre> | <pre>SELECT     CustomerID FROM     dbo.HUB_Customer</pre> |
| Rows in Table | 414 163                                                                                  | 55 000                                                     |
| Rows Returned | 55 000                                                                                   | 55 000                                                     |

In contrast, the researcher expected that the data vault model would perform worse than the dimensional model for higher category queries, that is, more complex queries. The data vault model would need to perform more join operations than the dimensional model in order to return the result set. Another aspect that needed to be considered was the fact that business rules in the dimensional model are implemented when the data is loaded into the data warehouse although this is not the case for data vault models. In the case of data vault models, the business rules are applied when the data is queried from the data vault. This would, in turn, also result in worse performing queries as compared to the dimensional model.

#### 4.7.4.4 Execute the experiment

This section describes the way in which the experiment was designed and executed. Technical detail in respect of the exact implementation of the experiment is described in the experiment protocol that may be found in Annexure M. The experiment was executed after the historic and incremental data had been loaded into the target data models. Several queries from each category, as defined in table 4.14 above, were written for the dimensional model. Table 4.14 also lists the number of distinct queries in each category that was used in the experiment. For each of the queries, a query was written to return the exact same set of data from the data vault model. The queries were confirmed to be equivalent by balancing the resulting datasets with each other. Date time stamps were recorded before and after each query execution. In addition, the duration of each query was calculated. This process was repeated 15 times in order to ensure that the results were reliable. All queries were executed sequentially and the SQL Server service was restarted before the execution of each query. This was to ensure that SQL Server's caching mechanisms did not interfere with the results.

#### 4.7.4.5 Results and analysis

Table 4.16 below presents the mean query duration in seconds by query category and target data model. The data vault performed better than the dimensional model in two of the five categories.

**Table 4.16 Mean query duration in seconds by category and target data model**

| Category | Duration_DM | Duration_DV |
|----------|-------------|-------------|
| 1        | 1.012       | 1.668       |
| 2        | 0.815       | 0.524       |
| 3        | 33.22       | 1.486       |
| 4        | 12.86       | 13.251      |
| 5        | 14.679      | 30.371      |
| All      | 8.525       | 8.664       |

The statistical analysis of the results is presented in the tables and graphs below. Table 4.17 to table 4.21 depicts the t-test for dependent samples results for each of the categories individually while table 4.22 depicts the same analysis for all the categories combined. The results are visually depicted through the use of box and whisker plots as depicted in figure 4.10 to figure 4.15.

The dimensional model performed statistically better than the data vault model in two of the four categories (category 1 and 5). The data vault model also performed statistically better than the dimensional model in two of the four categories (category 2 and 3). In the remaining category (category 4) there were no statistically significant differences as regards query performance. There were also no statistically significant differences in query performance for all queries across all categories.

**Table 4.17 Category 1 query results, t-test for dependent samples**

| Variable    | Category=1<br>t-test for Dependent Samples (Query Results)<br>Marked differences are significant at p < .05000 |         |    |        |               |        |    |          |                    |                     |
|-------------|----------------------------------------------------------------------------------------------------------------|---------|----|--------|---------------|--------|----|----------|--------------------|---------------------|
|             | Mean                                                                                                           | Std.Dv. | N  | Diff.  | Std.Dv. Diff. | t      | df | p        | Confidence 95.000% | Confidence +95.000% |
| Duration_DM | 1.012                                                                                                          | 0.689   |    |        |               |        |    |          |                    |                     |
| Duration_DV | 1.668                                                                                                          | 1.358   | 45 | -0.656 | 0.675         | -6.518 | 44 | 0.000000 | 0.453              | 0.858               |

**Table 4.18 Category 2 query results, t-test for dependent samples**

| Variable    | Category=2<br>t-test for Dependent Samples (Query Results)<br>Marked differences are significant at p < .05000 |         |    |       |               |       |    |          |                    |                     |
|-------------|----------------------------------------------------------------------------------------------------------------|---------|----|-------|---------------|-------|----|----------|--------------------|---------------------|
|             | Mean                                                                                                           | Std.Dv. | N  | Diff. | Std.Dv. Diff. | t     | df | p        | Confidence 95.000% | Confidence +95.000% |
| Duration_DM | 0.815                                                                                                          | 0.487   |    |       |               |       |    |          |                    |                     |
| Duration_DV | 0.524                                                                                                          | 0.185   | 45 | 0.291 | 0.317         | 6.161 | 44 | 0.000000 | -0.386             | -0.196              |

**Table 4.19 Category 3 query results, t-test for dependent samples**

| Variable    | Category=3<br>t-test for Dependent Samples (Query Results)<br>Marked differences are significant at p < .05000 |         |    |        |               |         |    |          |                    |                     |
|-------------|----------------------------------------------------------------------------------------------------------------|---------|----|--------|---------------|---------|----|----------|--------------------|---------------------|
|             | Mean                                                                                                           | Std.Dv. | N  | Diff.  | Std.Dv. Diff. | t       | df | p        | Confidence 95.000% | Confidence +95.000% |
| Duration_DM | 33.220                                                                                                         | 0.717   |    |        |               |         |    |          |                    |                     |
| Duration_DV | 1.486                                                                                                          | 0.110   | 15 | 31.734 | 0.677         | 181.538 | 14 | 0.000000 | -32.109            | -31.359             |

**Table 4.20 Category 4 query results, t-test for dependent samples**

| Variable    | Category=4<br>t-test for Dependent Samples (Query Results)<br>Marked differences are significant at p < .05000 |         |    |        |               |        |    |          |                    |                     |
|-------------|----------------------------------------------------------------------------------------------------------------|---------|----|--------|---------------|--------|----|----------|--------------------|---------------------|
|             | Mean                                                                                                           | Std.Dv. | N  | Diff.  | Std.Dv. Diff. | t      | df | p        | Confidence 95.000% | Confidence +95.000% |
| Duration_DM | 12.860                                                                                                         | 2.882   |    |        |               |        |    |          |                    |                     |
| Duration_DV | 13.251                                                                                                         | 10.497  | 30 | -0.391 | 13.371        | -0.160 | 29 | 0.874003 | -4.603             | 5.384               |

**Table 4.21 Category 5 query results, t-test for dependent samples**

| Variable    | Category=5<br>t-test for Dependent Samples (Query Results)<br>Marked differences are significant at p < .05000 |         |    |         |               |        |    |          |                      |                     |
|-------------|----------------------------------------------------------------------------------------------------------------|---------|----|---------|---------------|--------|----|----------|----------------------|---------------------|
|             | Mean                                                                                                           | Std.Dv. | N  | Diff.   | Std.Dv. Diff. | t      | df | p        | Confidence 95.000% - | Confidence +95.000% |
| Duration_DM | 14.679                                                                                                         | 14.034  |    |         |               |        |    |          |                      |                     |
| Duration_DV | 30.371                                                                                                         | 30.448  | 30 | -15.693 | 16.422        | -5.234 | 29 | 0.000013 | 9.561                | 21.825              |

**Table 4.22 All categories query results, t-test for dependent samples**

| Variable    | Category=All Categories<br>t-test for Dependent Samples (Query Results)<br>Marked differences are significant at p < .05000 |         |     |        |               |        |     |          |                      |                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------|---------|-----|--------|---------------|--------|-----|----------|----------------------|---------------------|
|             | Mean                                                                                                                        | Std.Dv. | N   | Diff.  | Std.Dv. Diff. | t      | df  | p        | Confidence 95.000% - | Confidence +95.000% |
| Duration_DM | 8.525                                                                                                                       | 11.596  |     |        |               |        |     |          |                      |                     |
| Duration_DV | 8.664                                                                                                                       | 17.615  | 165 | -0.139 | 14.724        | -0.121 | 164 | 0.903792 | -2.125               | 2.402               |

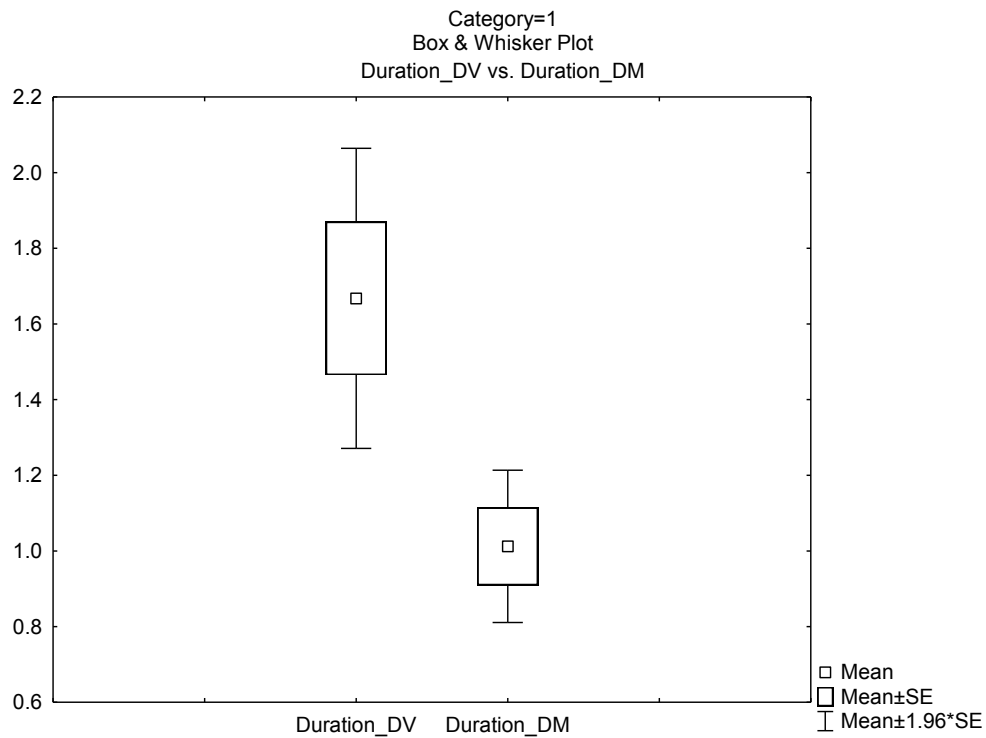


**Figure 4.10 Category 1 query results, box-and-whisker plot**

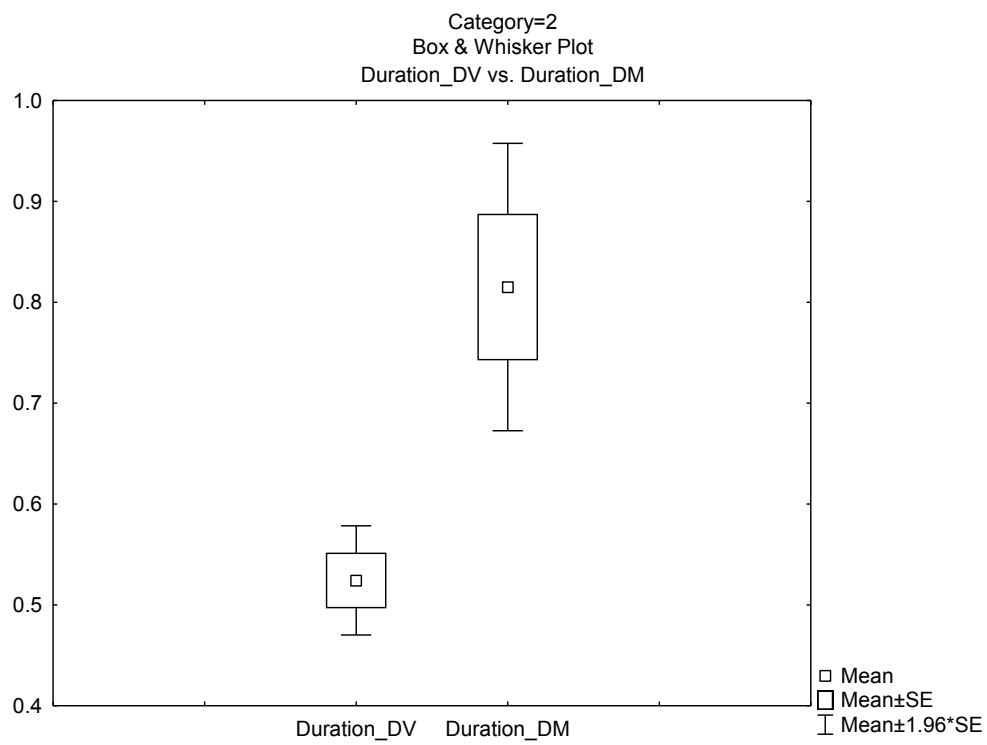


**Figure 4.11 Category 2 query results, box and whisker plot**

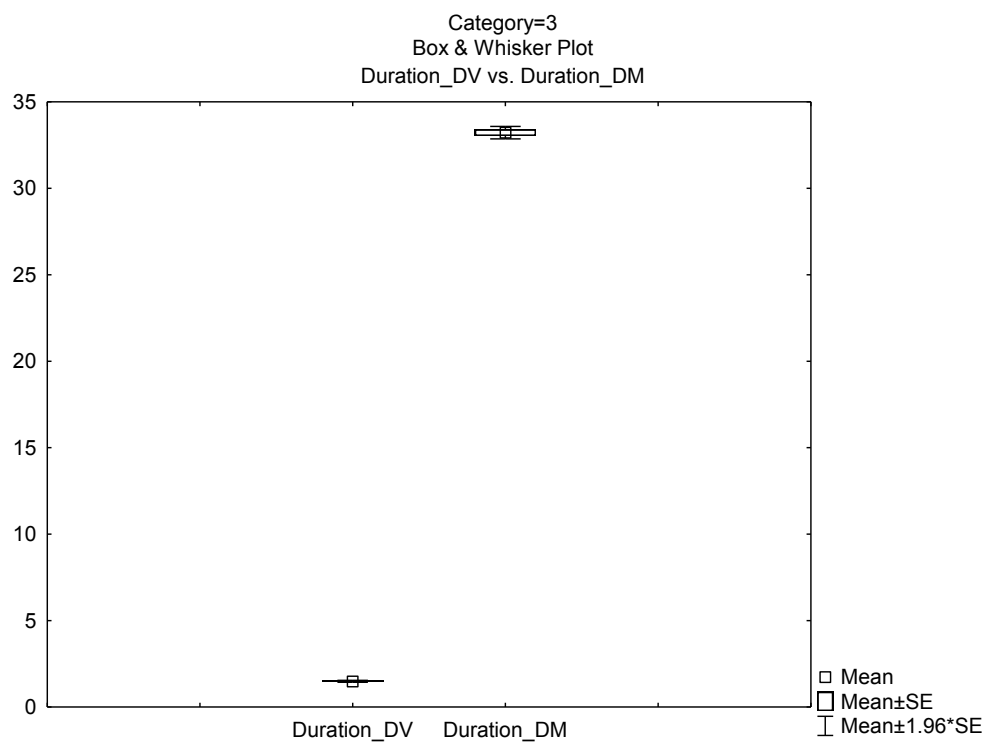


**Figure 4.12 Category 3 query results, box-and-whisker plot**

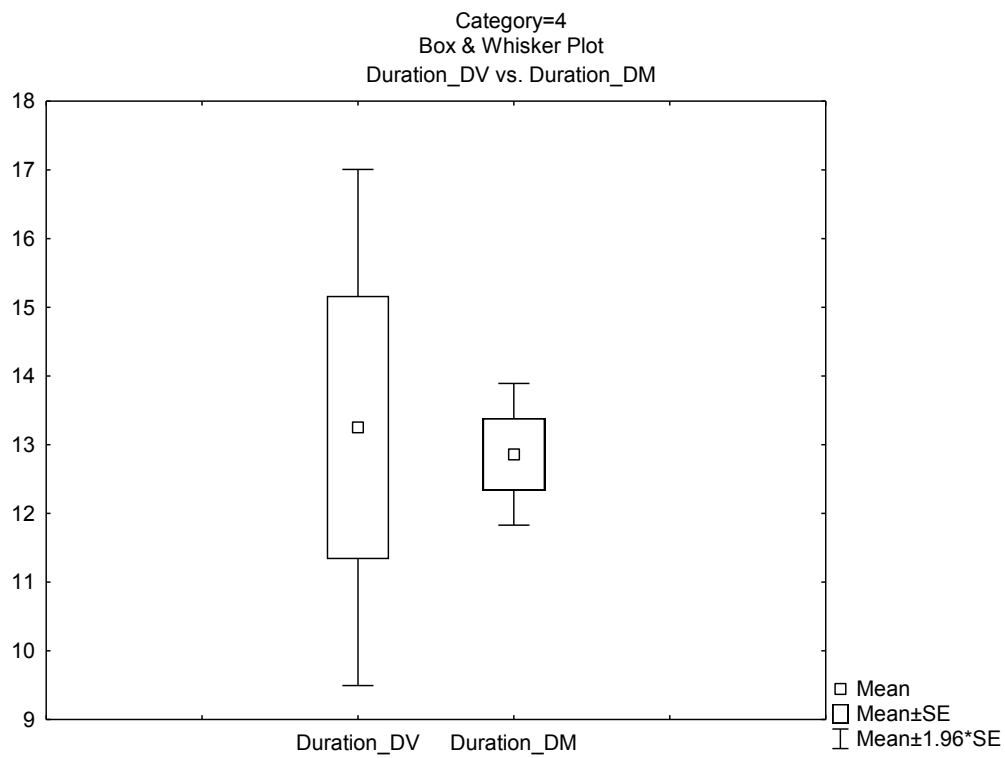


**Figure 4.13 Category 4 query results, box-and-whisker plot**

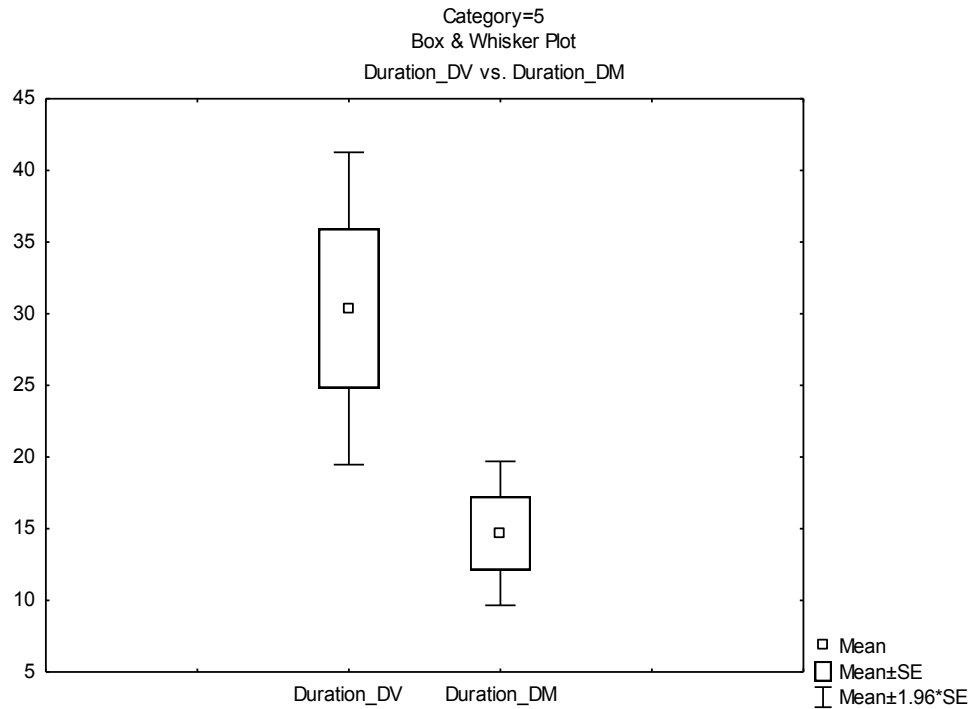


**Figure 4.14 Category 5 query results, box-and-whisker plot**

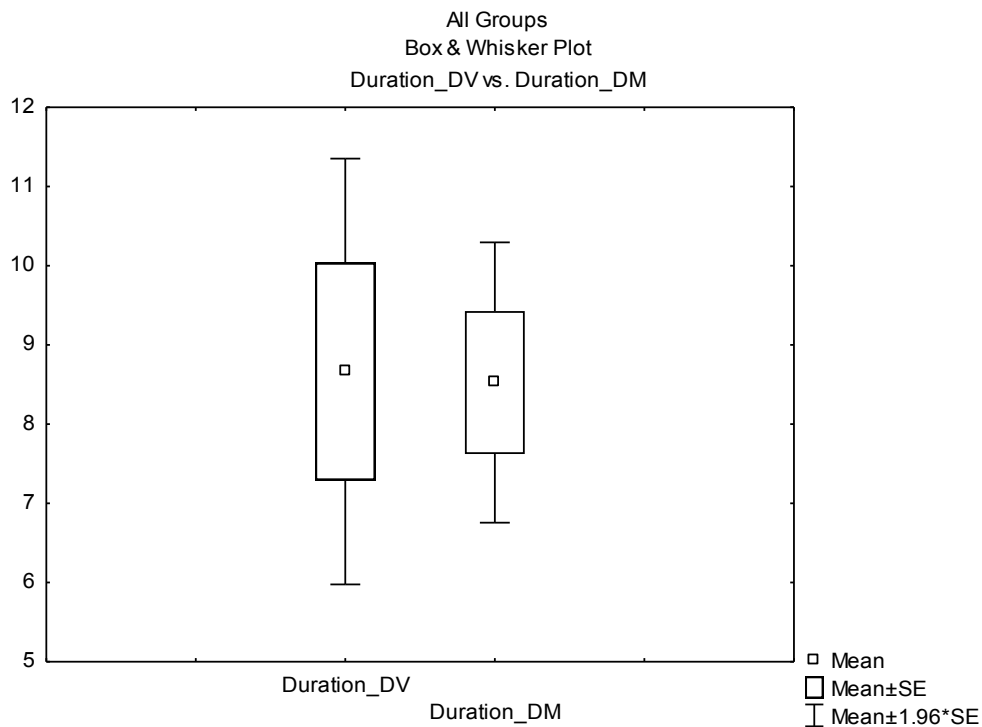


**Figure 4.15 All categories query results, box-and-whisker plot**

#### 4.7.4.6 Conclusion

The results from this experiment fail to support the hypothesis that the mean query execution time for queries to a data vault model is less than the mean query execution time for similar queries to the dimensional model. When each of the categories is considered in isolation, there are some statistically significant differences in respect of some of the categories. In this experiment category 5 queries constitute queries that are more representative examples of data warehouse queries than the other categories. For category 5 queries the results refutes the hypothesis and indicate that the mean query execution time for queries to a data vault model is greater than the mean query execution time for similar queries to the dimensional model.

#### 4.7.5 Storage size

The storage size experiment was the next experiment that was conducted.



#### **4.7.5.1 Define the research question**

This experiment is aimed at comparing the dimensional and the data vault models in terms of the storage space required. Despite the fact that storage space has become cheaper, the main consideration in terms of data warehousing is disk I/O. The bigger the volume of data that needs to be stored, the more disk I/Os are required either to write the data to disk or to read the data from disk.

#### **4.7.5.2 Formulate the hypothesis**

The hypothesis for this experiment is formulated as follow:

*The mean storage space required for data in a data vault model is less than the mean storage space required for the same data in a dimensional model.*

The storage space required to store a set of data in a relational database is determined by two components, namely, data size and index size.

#### **4.7.5.3 Predict the results**

The researcher considered the following aspects when predicting the outcome of the experiment:

- The data vault model has less redundant data than the dimensional model. However, this is true for the data in dimension tables only. Higher level attributes in dimension tables are duplicated because the tables are denormalised in order to improve query performance. The data vault model reduces duplication by normalising the tables while the data contained in a single dimension table may be contained in several hub entities, each with some satellite entities and link entities that model the relationship between these hub entities. These link entities also have satellite entities attached to them. There is usually data duplication in satellite entities, but to a lesser degree than in dimension tables.
- Data redundancy in dimensional models occurs in dimension tables only. However, the storage space that would be saved by normalising these tables would have a relatively small impact on the overall size of a dimension table, as the bulk of the storage space used by a dimensional model may be attributed to fact tables (Kimball *et al.*, 2008:267).
- Type 2 slowly changing dimension also causes data duplication for those attributes that do not change. As an example, consider a dimension table with 50 attributes. If one of these attributes is a type 2 attribute and it changes, a new record is created and the other 49 attributes are duplicated. Data vault models track changes to attributes in

satellite entities. One of the data vault model design steps involves normalising satellite entities by rate of changes (Linstedt & Graziano, 2011:118). Thus, the implication is that not all contextual data is duplicated when a single attribute changes.

- The data vault model has more auditing fields than the dimensional model. The mandatory auditing fields that are part of the data vault model are presented in table 4.23 below. The dimensional model, as implemented in this research project, also has equivalent structures with which to load either date timestamp and load end data timestamp in the date inserted and date updated fields, or start and end date fields for type 2 SCDs. However, the difference is that the data vault model results in a higher number of tables than the dimensional model, and these fields are contained in each of these tables.

**Table 4.23 Mandatory auditing fields**

| Entity    | Mandatory Auditing data       | Data Type    |
|-----------|-------------------------------|--------------|
| Hub       | Hub sequence number           | INT / BIGINT |
|           | Load date timestamp           | DATETIME2(0) |
|           | Record source                 | VARCHAR(10)  |
| Link      | Multiple hub sequence numbers | INT / BIGINT |
|           | Load date timestamp           | DATETIME2(0) |
|           | Record source                 | VARCHAR(10)  |
| Satellite | Hub or link sequence number   | INT / BIGINT |
|           | Load date timestamp           | DATETIME2(0) |
|           | End date timestamp            | DATETIME2(0) |
|           | Record source                 | VARCHAR(10)  |

- Both the dimensional model and the data vault model follow similar indexing strategies. Index size differences between the dimensional model and the data vault model for all but the very large tables are negligible (Microsoft, 2012). However, on tables with a large number of rows, such as fact tables or transactional link tables, differences in indexing strategies may have a significant effect. Using the approach defined by Microsoft (2012) the size of a non-unique, nonclustered index on a single INT field for F\_Trade or TLNK\_Trade will be 548MB. This is based on the number of rows in these tables. In view of the fact that the data vault model does not have the equivalent of a date dimension, TLNK\_Trade will have one less integer field to index as compared to F\_Trade and, therefore, it requires 548MB less index storage space.

Based on the above considerations the researcher predicted that the mean storage space required for data in a data vault model would be less than the mean storage space required for

the same data in a dimensional model. The researcher expected index storage space to be the major contributor to this difference.

#### 4.7.5.4 Execute the experiment

The storage space experiment was executed in conjunction with the incremental load performance experiment. After each incremental load, the data size and index size of each of the target databases were recorded. The reader will recall from section 4.7.3.4 above that the incremental data was loaded for 30 consecutive days. This process was repeated three times, resulting in 90 data points.

#### 4.7.5.5 Results and analysis

The results of the storage size experiment are summarised in table 4.24 below. The table presents the mean, minimum and maximum data index and total size for each of the target data models in megabytes (MB). The growth was calculated as  $\frac{Max}{Min} - 1$ . These results are aggregated across the 30 incremental loads and the three repetitions of the experiment. The mean size of the dimensional model is greater than the mean size of the data vault model in terms of data and index size. These differences are confirmed as statistically significant by the t-test for dependent samples that were conducted for each. The results for the t-test for dependent samples are depicted below in table 4.25 to table 4.27 and also visually in the box and whisker plots in figure 4.16 to figure 4.17.

**Table 4.24 Storage size experiment results in MB**

|        | DM    |       |       | DV    |       |       |
|--------|-------|-------|-------|-------|-------|-------|
|        | Data  | Index | Total | Data  | Index | Total |
| Mean   | 1 727 | 2 071 | 3 799 | 1 713 | 1 850 | 3 564 |
| Min    | 834   | 1 057 | 1 891 | 760   | 846   | 1 606 |
| Max    | 2 661 | 3 147 | 5 808 | 2 723 | 2 915 | 5 639 |
| Growth | 219%  | 198%  | 207%  | 258%  | 245%  | 251%  |

**Table 4.25 Data size results, t-test for dependent samples**

| Variable    | t-test for Dependent Samples (Size Results)<br>Marked differences are significant at p < .05000 |         |    |        |               |       |    |          |                     |                     |
|-------------|-------------------------------------------------------------------------------------------------|---------|----|--------|---------------|-------|----|----------|---------------------|---------------------|
|             | Mean                                                                                            | Std.Dv. | N  | Diff.  | Std.Dv. Diff. | t     | df | p        | Confidence -95.000% | Confidence +95.000% |
| DataSize_DM | 1727.392                                                                                        | 545.761 |    |        |               |       |    |          |                     |                     |
| DataSize_DV | 1713.278                                                                                        | 588.279 | 90 | 14.114 | 43.637        | 3.140 | 89 | 0.002290 | 5.184               | 23.044              |

**Table 4.26 Index size results, t-test for dependent samples**

| Variable     | t-test for Dependent Samples (Size Results)<br>Marked differences are significant at $p < .05000$ |          |    |         |               |         |    |          |                     |                     |
|--------------|---------------------------------------------------------------------------------------------------|----------|----|---------|---------------|---------|----|----------|---------------------|---------------------|
|              | Mean                                                                                              | Std.Dv.  | N  | Diff.   | Std.Dv. Diff. | t       | df | p        | Confidence -95.000% | Confidence +95.000% |
| IndexSize_DM | 2071.218                                                                                          | 627.0279 |    |         |               |         |    |          |                     |                     |
| IndexSize_DV | 1850.355                                                                                          | 620.965  | 90 | 220.883 | 6.067         | 345.372 | 89 | 0.000000 | 219.613             | 222.154             |

**Table 4.27 Total size results, t-test for dependent samples**

| Variable     | t-test for Dependent Samples (Size Results)<br>Marked differences are significant at $p < .05000$ |          |    |         |               |        |    |          |                     |                     |
|--------------|---------------------------------------------------------------------------------------------------|----------|----|---------|---------------|--------|----|----------|---------------------|---------------------|
|              | Mean                                                                                              | Std.Dv.  | N  | Diff.   | Std.Dv. Diff. | t      | df | p        | Confidence -95.000% | Confidence +95.000% |
| TotalSize_DM | 3798.611                                                                                          | 1172.784 |    |         |               |        |    |          |                     |                     |
| TotalSize_DV | 3563.613                                                                                          | 1209.244 | 90 | 234.998 | 36.603        | 60.907 | 89 | 0.000000 | 227.331             | 242.664             |

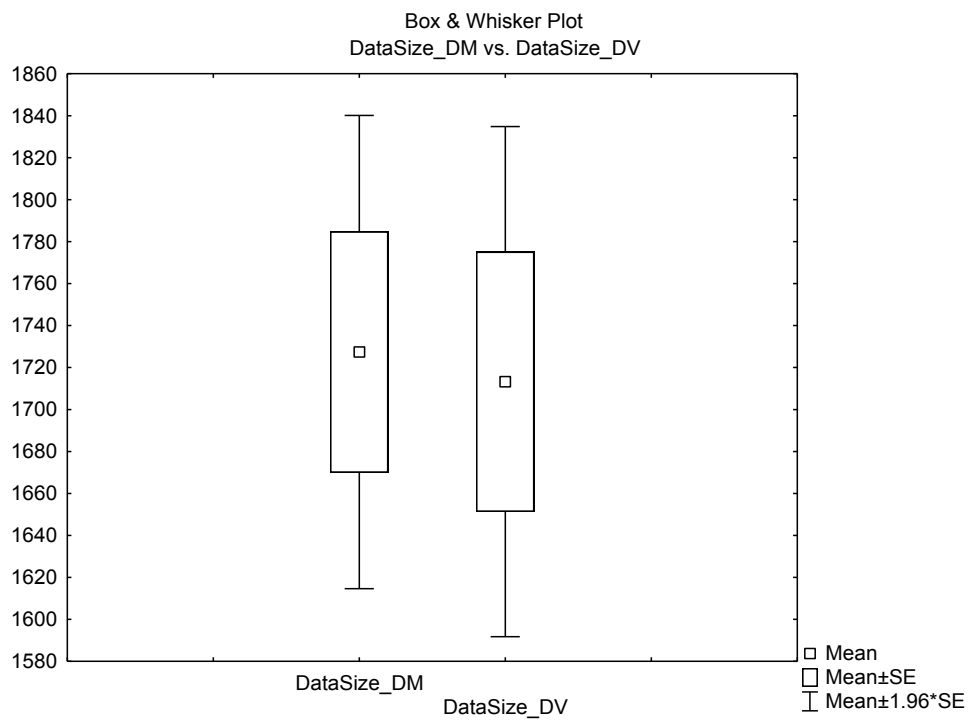


**Figure 4.16 Data size results, box-and-whisker plot**

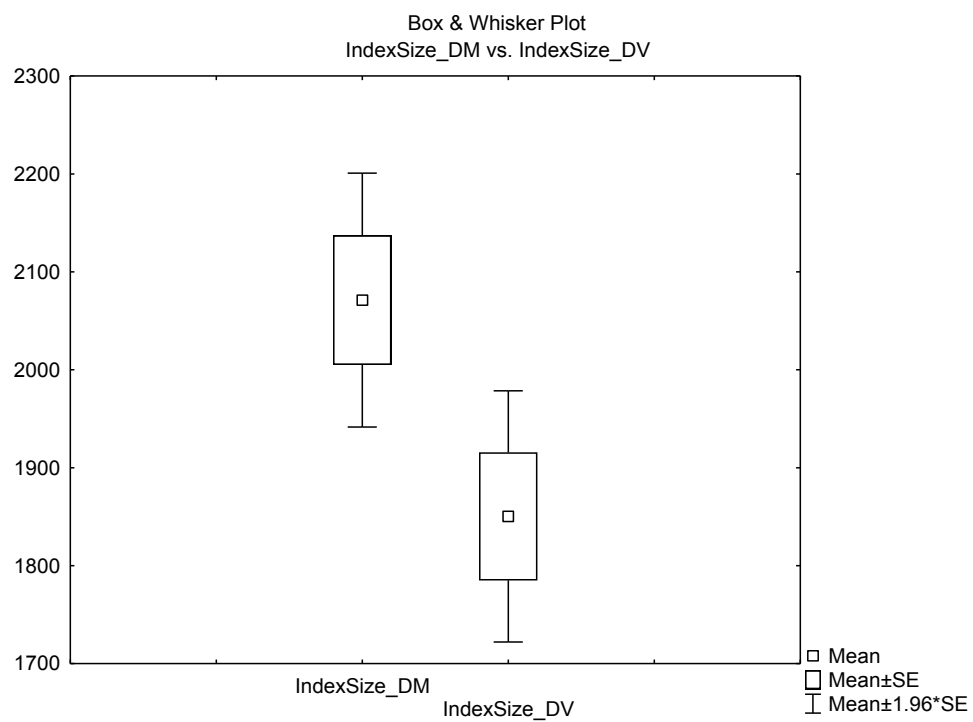


**Figure 4.17 Index size results, box-and-whisker plot**

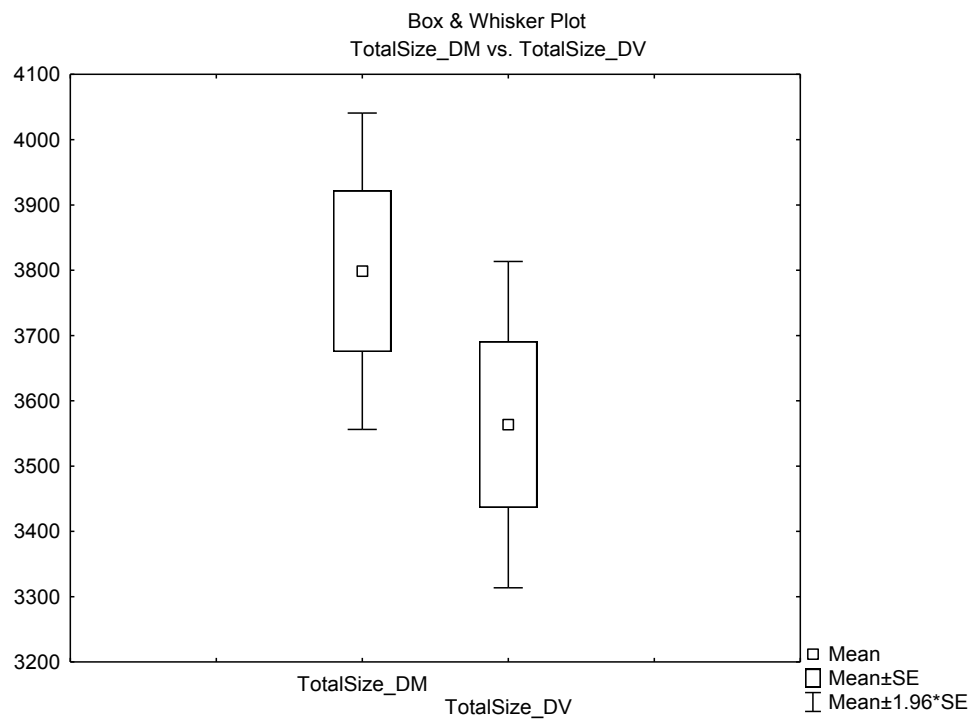


**Figure 4.18 Total size results, box-and-whisker plot**

These results support the hypothesis that the mean storage space required for data in a data vault model is less than the mean storage space required for the same data in a dimensional model. However, this is true only when the data from 40 days is loaded into both target models, as was the case in this experiment.

It would appear that the maximum and growth values in table 4.24 above indicate that this would not be true if more data were loaded into the target data models, for example data from five years. Although the index and total maximum storage size for the data vault model were still less than that of the dimensional model, the growth of the data vault model over the 30 incremental loads was higher than that of the dimensional model. In addition, the maximum data storage size and the growth of the data storage size of the data vault model were higher than that of the dimensional model. This is visually highlighted by the scatterplots in figure 4.19 below. Figure 4.19 depicts that the data storage size for the data vault model exceeded the data storage size for the dimensional model after 19 days of incremental data had been loaded. As shown in figure 4.20 the index storage size for both models was growing at similar rate. However, the data vault model was using less storage space than the dimensional model. Although the total storage size for the data vault model was still less than that of the dimensional model, figure 4.21 indicates that this may change as more data is loaded into the target data models.

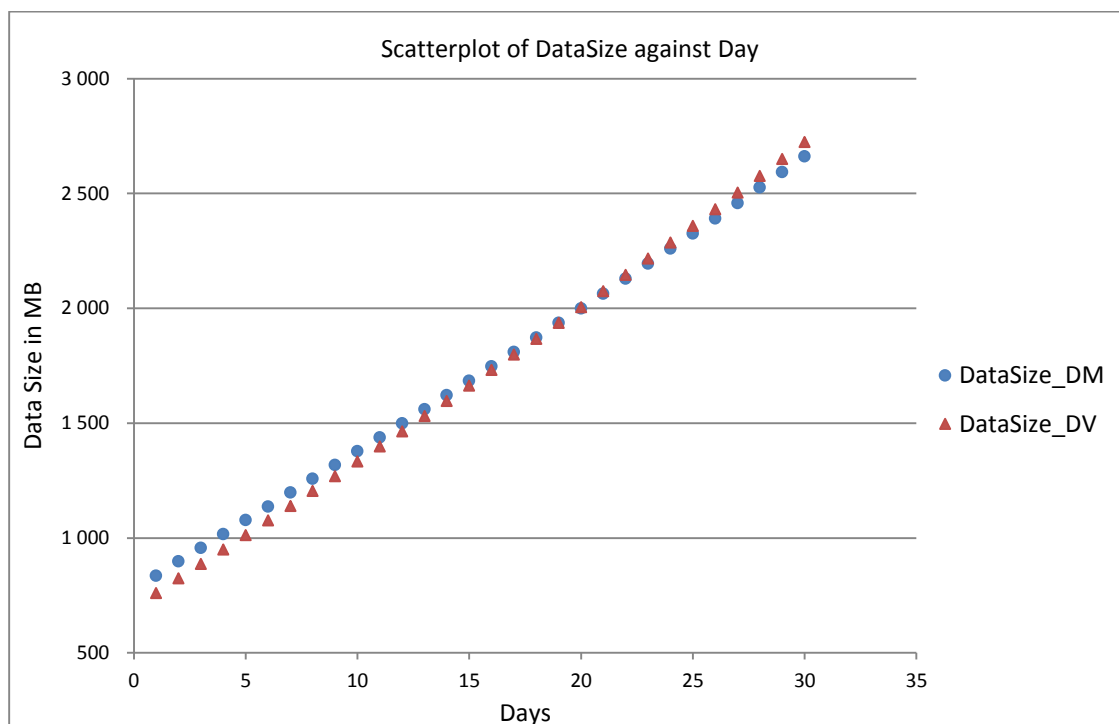


**Figure 4.19** Size results, scatterplot of data size over time

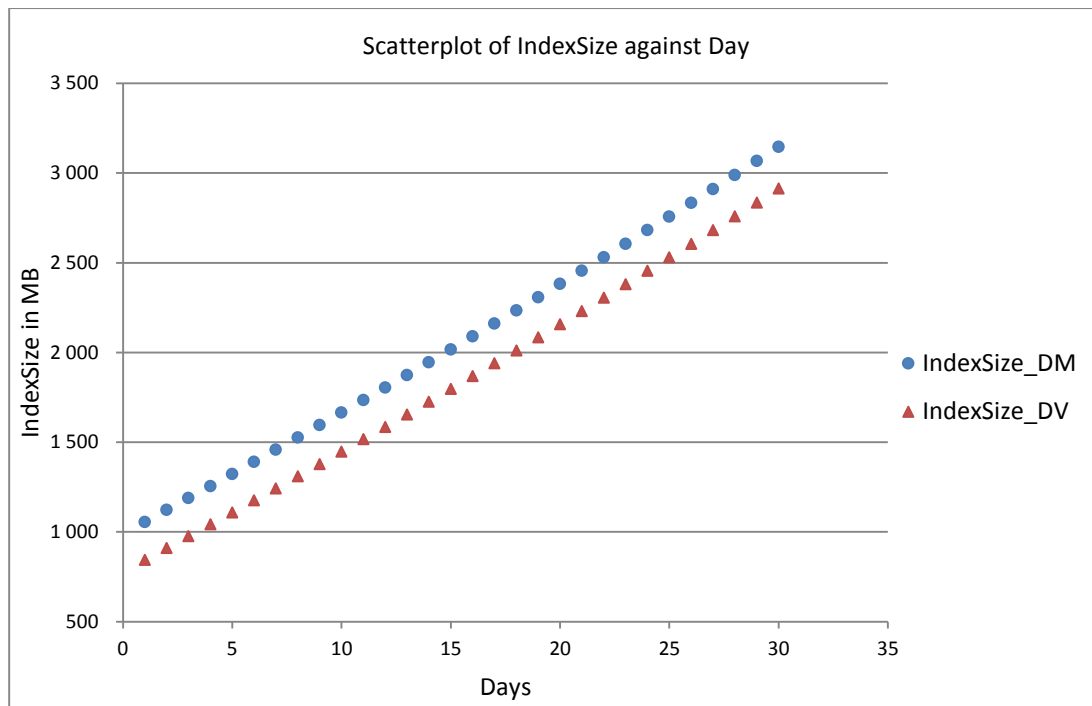


**Figure 4.20** Size results, scatterplot of index size over time

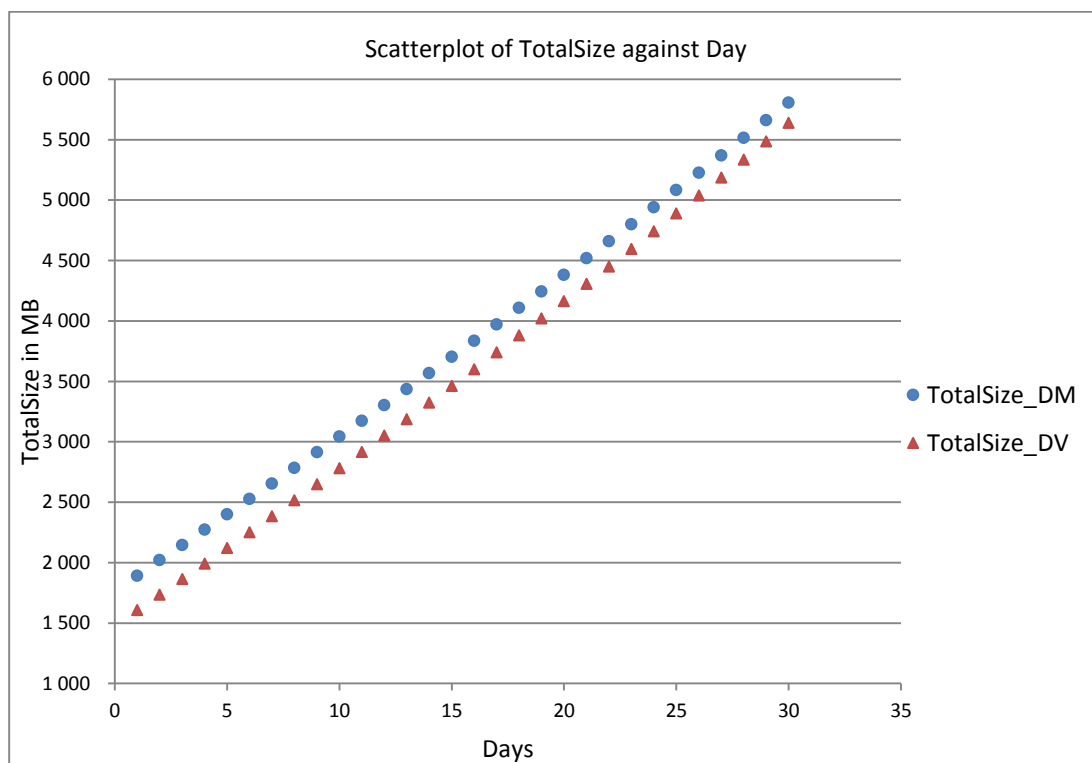


**Figure 4.21** Size results, scatterplot of total size over time

A time-series algorithm was used to forecast the storage sizes for each of the data models after 100 days of additional incremental loads. The result of this algorithm is depicted in figure 4.22 below and summarised in table 4.28 below.

**Table 4.28 Storage predictions after an additional 100 incremental loads**

| Measure            | Predicted Size (GB) |           |       |
|--------------------|---------------------|-----------|-------|
|                    | Dimensional (DM)    | Data (DV) | Vault |
| Data storage size  | 10.19               | 11.78     |       |
| Index storage size | 12.92               | 12.61     |       |
| Total storage size | 23.11               | 24.39     |       |

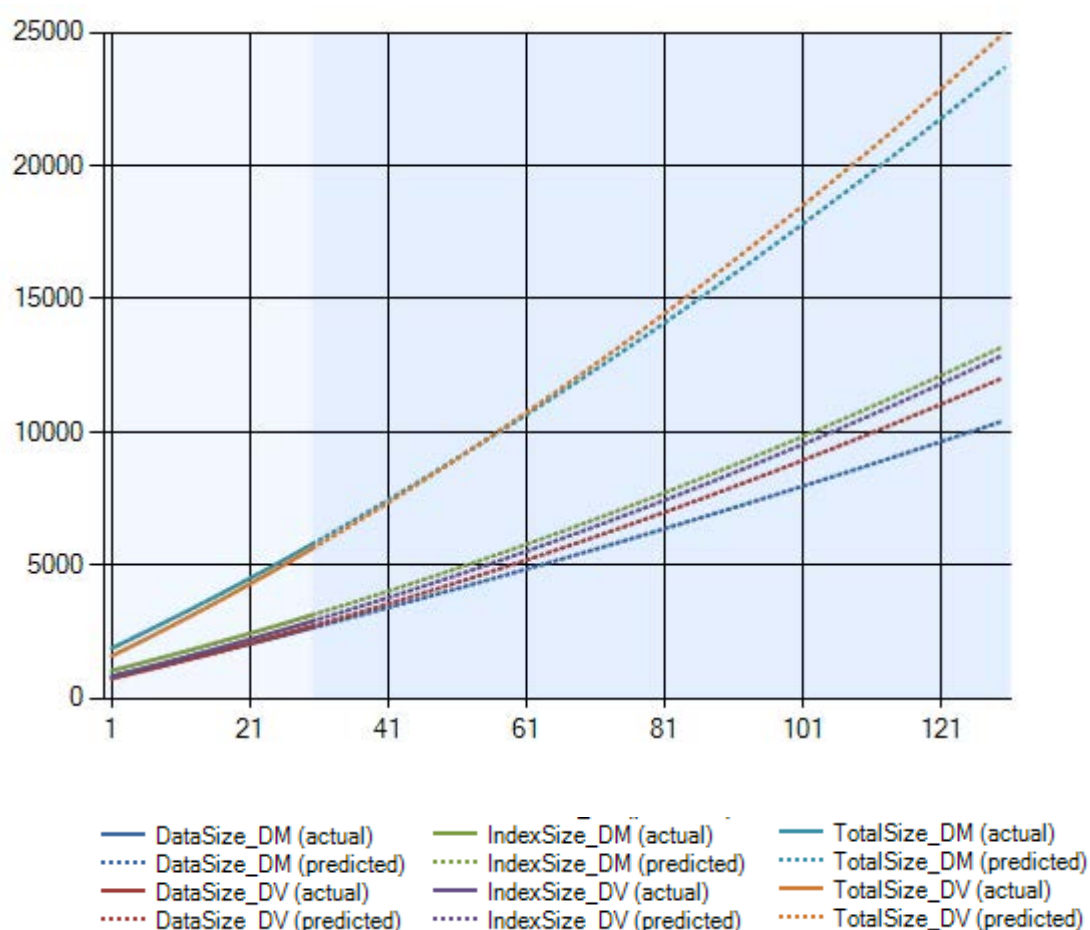


**Figure 4.22 Storage predictions after an additional 100 incremental loads**



#### **4.7.5.6 Conclusion**

The analysis above indicate that, despite the fact that the hypothesis for this particular experiment is supported, it would not be true if the experiment were conducted with a dataset that spanned a longer time period, for example 100 days. It may, thus, safely be concluded that the mean storage space required for data in a data vault model is not less than the mean storage space required for the same data in a dimensional model under all circumstances.

#### **4.7.6 Flexibility**

The flexibility experiment was the next experiment that was conducted.

##### **4.7.6.1 Define research question**

This experiment was aimed at comparing the dimensional and data vault models in terms of flexibility. Data warehouse requirements change continuously (Kimball, 2001). This includes changes to existing objects in the model as well as the addition of new objects. The aim of this experiment was to test the effect that changes in the existing requirements or additional requirements would have on each of the target data models.

The effect of these changes was measured in terms of the impact they had on existing queries. The researcher classified as unaffected queries any queries that executed without any error messages and that returned exactly the same results before and after any changes/additions to the target data model. Queries that either returned error messages or that returned a different result set after changes/additions to the target data model were classified as affected queries.

##### **4.7.6.2 Formulate the hypothesis**

The hypothesis for this experiment was formulated as follow:

*The number of existing queries that will be affected after changes/additions to the data vault model are made in order to meet changing user requirements is less than the number of existing queries that will be affected on a dimensional model after changes/additions to the model in order to meet the same set of requirements.*

#### **4.7.6.3 Predict the results**

The flexibility of the link entities in the data vault model was the major consideration when predicting the results of this experiment. The flexibility of the link entities lies in the fact that link entities always represent many-to-many relationships between business keys as opposed to one-to-many or many-to-one relationships (Linstedt & Graziano, 2010:57). Another characteristic of the data vault model is that requirements for additional data usually result in additional tables in the data vault model rather than changes to the existing tables.

Based on the above characteristics of the data vault model, the researcher predicted that the mean number of existing queries that would be affected after changes/additions to the data vault model had been made in order to meet changing user requirements would be less than the mean number of existing queries that would be affected on a dimensional model after changes/additions to the model in order to meet the same set of requirements.

#### **4.7.6.4 Execute the experiment**

This section will first discuss the design of the experiment and then the execution of the experiment.

##### **Experiment Design**

Three changes to the user requirements, as detailed in section 4.5, were introduced for the purposes of this experiment. The two target data models were adapted in order to meet these new requirements. The queries that were used in the query performance experiment were executed against the updated models. Successes or failures were recorded. Three categories of changes were tested:

1. Change to existing requirements: change a 1:1 or more relationship into a 1:0 or more relationship.
2. Change to existing requirements: change a 1:1 or more relationship into a many:many relationship.
3. Add new requirements.

Below is one example of each of the categories of changes that were used in the experiment.

The changed/additional requirements were as follow:

1. Change a 1:1 or more relationship into a 1:0 or more relationship: Customer information was stored in the data warehouse only if an account for that customer existed. Both the existing dimensional model and data vault model cater for this need. However, a change in requirements would mean that customer information should then be stored in the data warehouse for marketing purposes, irrespective of whether the customer has a trading account. The new requirement is illustrated in figure 4.23 below while figure 4.24 depicts the original requirement for reference purposes.

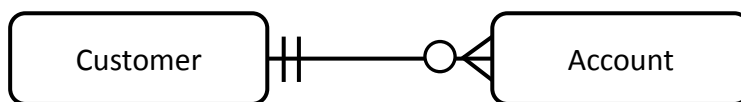


**Figure 4.23 Changed customer account relationship**

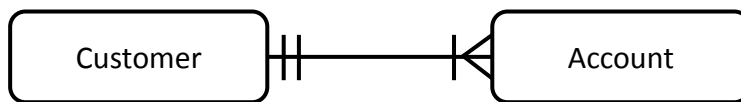


**Figure 4.24 Original customer account relationship**

2. Change a 1:1 or more relationship into a many:many relationship. The data in the data warehouse came from a single source system, TPCE. However, data from an additional marketing database needed to be added to the data warehouse. Both source systems contained data on companies and on the industry to which the company belonged. However, the marketing database grouped companies into industries that were not always the same as the industry in the TPC database. Figure 4.25 below illustrates the new requirement while figure 4.26 illustrates the original requirement for reference purposes.

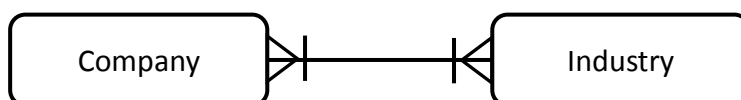


**Figure 4.25 Changed industry company relationship**

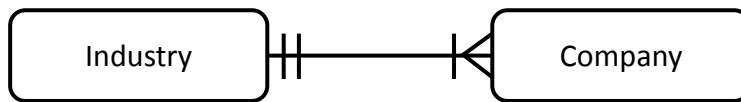


**Figure 4.26 Original industry company relationship**

3. Add new requirements: The final requirements change involved brokers. Brokers who traded on behalf of the trading company were employed by the trading company. However, their business model had since changed and they were making use of contracting brokers. In order to incorporate this change into the data warehouse, the notion of “Broker Type” had to be included in the data warehouse.

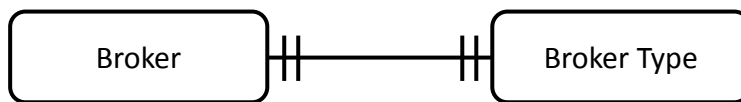


**Figure 4.27 New broker type requirement**

## Experiment implementation

Each of the above changes required changes to the target data model. In order to handle the change in the customer account relationship in the dimensional model the D\_CustomerAccount dimension was split into a customer dimension, D\_Customer, and an account dimension, D\_Account. In addition, a dimension foreign key to D\_Account also had to be added to F\_Trade. The above change is illustrated in figure 4.28 below. Note that two existing tables, D\_Customer and F\_Trade, had to be changed and an additional table, D\_Account, had to be introduced to accommodate the change in requirements. Note also that the design in figure 4.1 may have been implemented to meet the original requirement as well. However, that would have meant that it would have been possible only to associate customers to their related accounts either through the fact table or by introducing a customer outrigger to D\_Account.

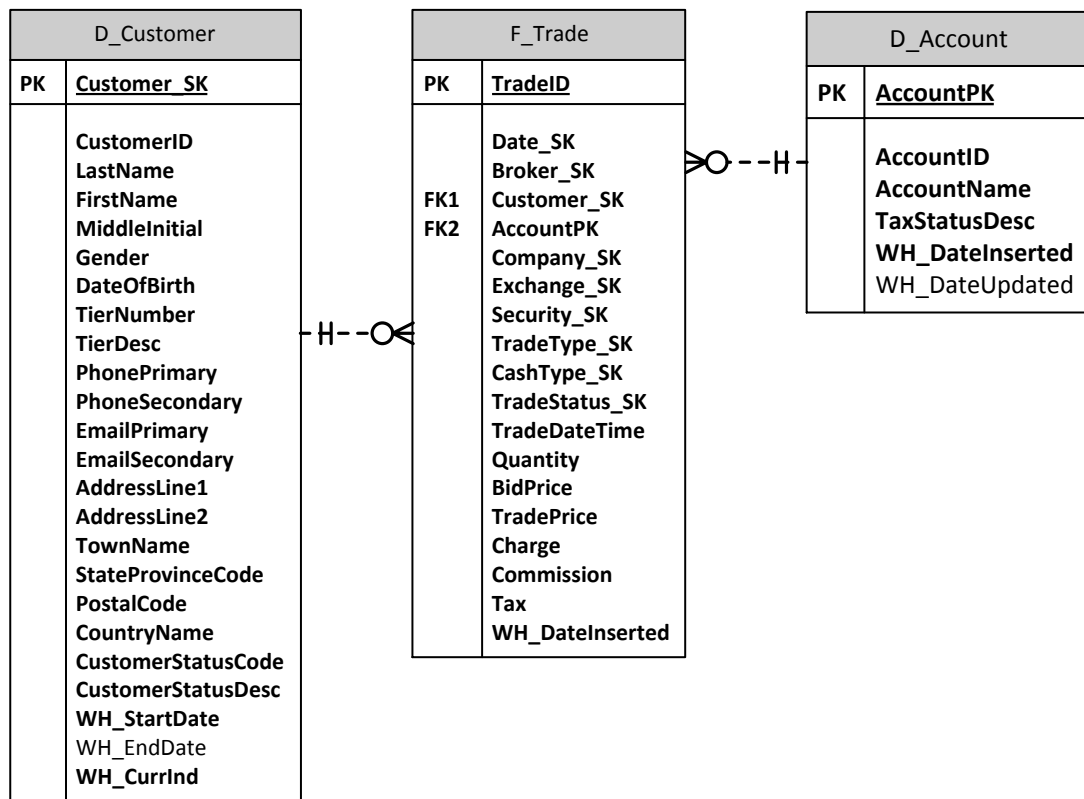


**Figure 4.28 Customer account relationship change in dimensional model**

No change was required to the data vault model in order to meet the changing requirements. All the business key relationships in the data vault were modelled through link entities which effectively resulted in a many-to-many relationship between the two business keys (Linstedt & Graziano, 2011:79). As a result, a change in relationship between the account and customer hub entities did not require a change to the existing data model. It should be noted that, unlike the dimensional model, there was no other alternative to modelling the customer and account data in the dimensional model.

The second requirement change to be considered, namely, the many-to-many relationship between company and industry, also required a change to the dimensional model. Two design alternatives were identified for this requirement. As a first alternative, it was possible to remove the industry attributes from the company dimension and place them into their own industry dimension. A bridge table could then be used to model the many-to-many relationship between the two dimensions. This design is illustrated in figure 4.29 below. Note that a source field was inserted into the bridge table. This allowed for companies to be related to a specific industry based on the source system (either TPCE or marketing). This field could also have been called ReportingDepartment. As a result of this design, one table was changed while two new tables were introduced. A second alternative is illustrated in figure 4.30. A second set of industry

attributes was added to the company dimension. This also allowed for a company to be reported on either by industry according to the TPCE source system or according to the marketing source system/department. As a result of this change, two new fields were added to an existing table.

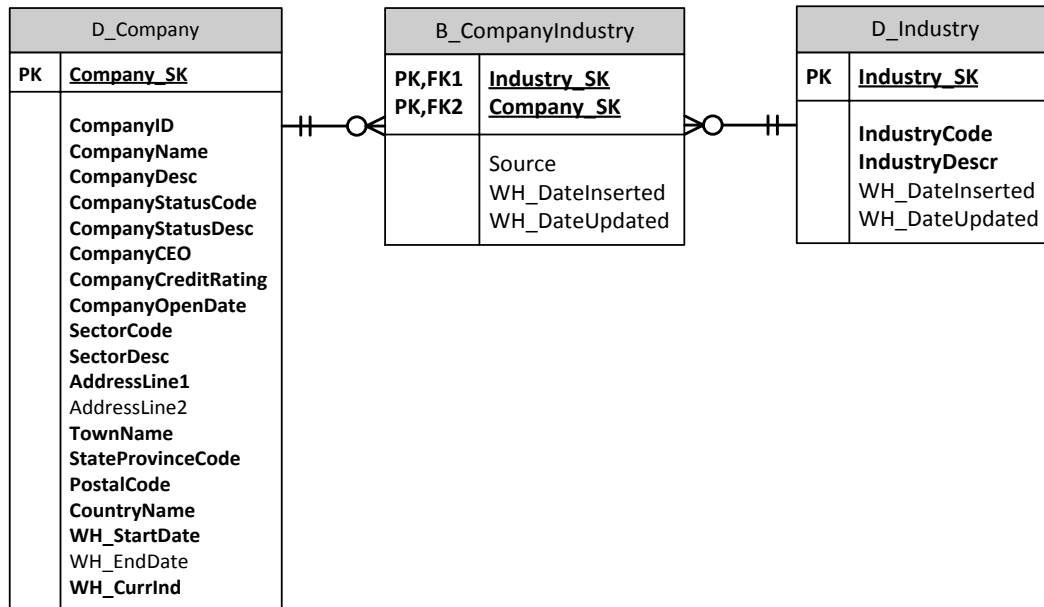


**Figure 4.29** Many-to-many relationship between company and industry with bridge table

| D_Company |                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PK        | <u>Company_SK</u>                                                                                                                                                                                                                                                                                                                                                                                 |
|           | CompanyID<br>CompanyName<br>CompanyDesc<br>CompanyStatusCode<br>CompanyStatusDesc<br>CompanyCEO<br>CompanyCreditRating<br>CompanyOpenDate<br>IndustryCode<br>IndustryDesc<br>IndustryCodeMarketing<br>IndustryDescMarketing<br>SectorCode<br>SectorDesc<br>AddressLine1<br>AddressLine2<br>TownName<br>StateProvinceCode<br>PostalCode<br>CountryName<br>WH_StartDate<br>WH_EndDate<br>WH_CurrInd |

**Figure 4.30 Many-to-many dimensional relationship between company and industry**

Changes to the data vault model were also required in order to meet the many-to-many relationship requirement between company and industry. The industry to which the company relates was contained in the company satellite entity. Linstedt and Graziano recommend normalising satellites by source system (2011:120). In view of the fact that the new industry data came from a separate marketing source system the industry code was put into a separate company satellite, SAT\_MarketingCompany. As a result of this change, an additional table was added to the data vault model. However, no existing tables were modified.

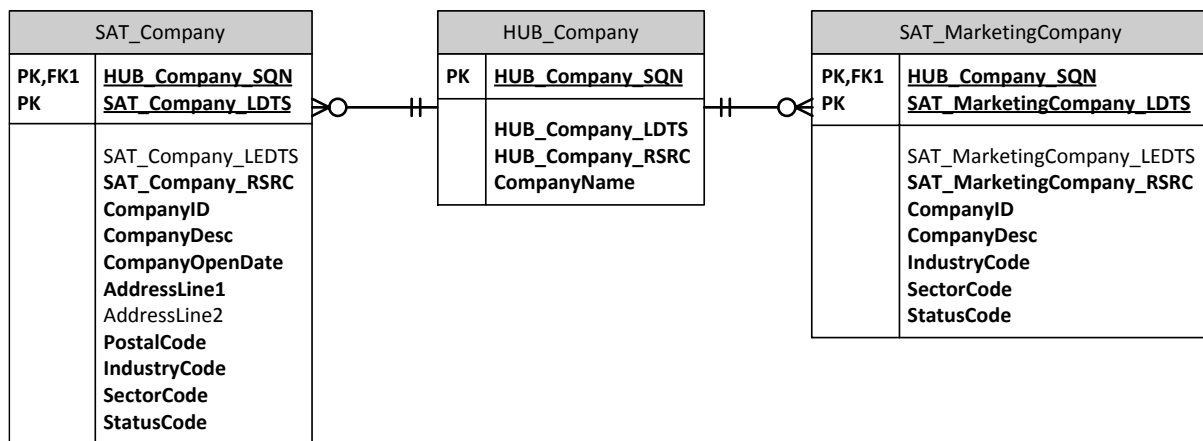


**Figure 4.31 Many-to-many data vault relationship between company and industry**

For the broker type requirement change, a simple change to the dimensional model was required with A BrokerType attribute being added to D\_Broker – see figure 4.32 below. For reference purposes, the original D\_Broker dimension is depicted in figure 4.33. In effect, a single field was added to a single dimension table in order to incorporate the changed requirement into the dimensional model.

| D_Broker |                                                                                                                   |
|----------|-------------------------------------------------------------------------------------------------------------------|
| PK       | <u>Broker_SK</u>                                                                                                  |
|          | BrokerID<br>BrokerName<br>BrokerStatusCode<br>BrokerStatusDesc<br>BrokerType<br>WH_DateInserted<br>WH_DateUpdated |

**Figure 4.32 Broker-type change**



| D_Broker |                                                                                                     |
|----------|-----------------------------------------------------------------------------------------------------|
| PK       | <u>Broker_SK</u>                                                                                    |
|          | BrokerID<br>BrokerName<br>BrokerStatusCode<br>BrokerStatusDesc<br>WH_DateInserted<br>WH_DateUpdated |

**Figure 4.33 Original D\_Broker dimension**

Several changes to the data vault model were required in order to meet the BrokerType requirement change. An additional hub entity, HUB\_BrokerType, was introduced into the data vault model while a link entity, LNK\_BrokerTypeBroker, was used to model the relationship between HUB\_Broker and HUB\_BrokerType. LNK\_BrokerTypeBroker also incorporated a satellite entity. The amended section of the data vault model is depicted in figure 4.34 below. Although three additional tables were added to the data vault model, none of the existing tables changed.

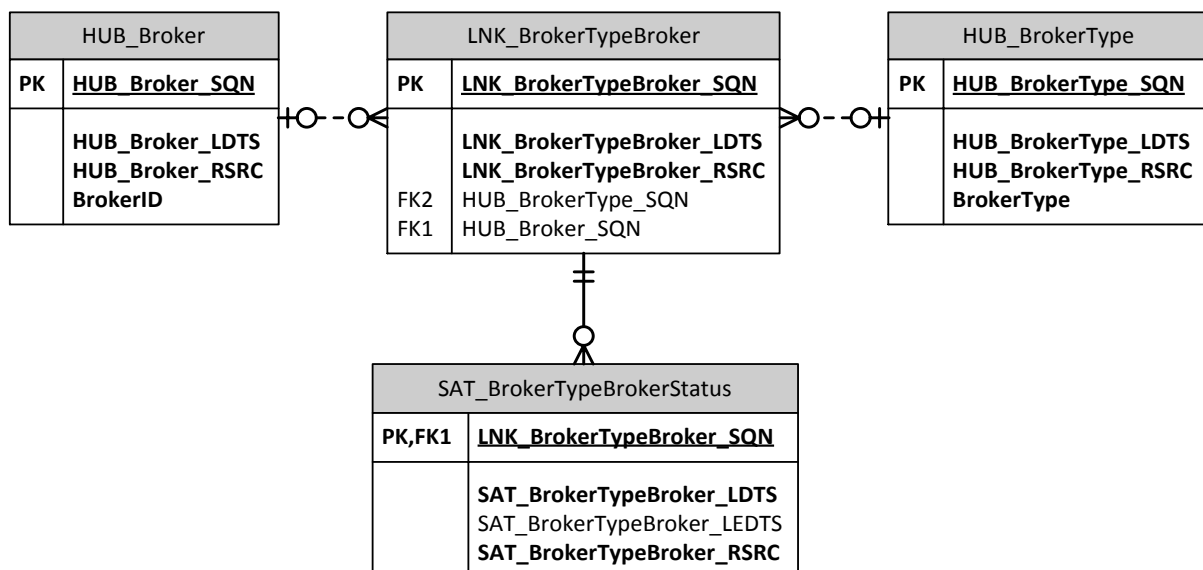


**Figure 4.34 Broker type change to data vault model**

#### 4.7.6.5 Results and analysis

Each of the queries that was executed in the query performance experiment were evaluated to determine whether they had been affected after the changes to the two data models, as

described above, had been implemented. The summary of the results is presented in table 4.29 below. From these results it is evident that the data vault model is more resilient to changes in business requirements when compared to the dimensional model, with only a single query out of the eleven not working for the data vault model.

**Table 4.29 Flexibility experiment results**

| Category       | Primary Table               | Dimensional Model | Data Vault Model |
|----------------|-----------------------------|-------------------|------------------|
| 1              | Company                     | Affected          | Unaffected       |
| 1              | CustomerAccount             | Affected          | Unaffected       |
| 1              | Security                    | Unaffected        | Unaffected       |
| 2              | Company                     | Affected          | Affected         |
| 2              | CustomerAccount             | Unaffected        | Unaffected       |
| 2              | Security                    | Unaffected        | Unaffected       |
| 3              | CustomerBroker              | Unaffected        | Unaffected       |
| 4              | DailyMarketPerformance      | Unaffected        | Unaffected       |
| 4              | Trade                       | Affected          | Unaffected       |
| 5              | QuarterlyCompanyPerformance | Unaffected        | Unaffected       |
| 5              | Trade                       | Unaffected        | Unaffected       |
| Total Affected |                             | 4                 | 1                |

#### 4.7.6.6 Conclusion

From the results, it is clear that the number of existing queries that would be affected after changes/additions to the data vault model had been made in order to meet changing user requirements is less than the number of existing queries that would be affected on a dimensional model after similar changes/additions to the model in order to meet the same set of requirements. The hypothesis is, therefore, supported.

## 4.8 Summary

This chapter started with an overview of the experiments that were conducted. The chapter described the environment in which the experiments were conducted in terms of both the software and the configuration of the software. The use of the TPC-E dataset was motivated and the dataset was described. Details of the two target data models, the dimensional model and the data vault model, were then provided. Finally, the experiments were discussed. This discussion included the aim of each of the experiments as well as the hypothesis/hypotheses

that was/were formulated. The outcomes of the experiments were predicted and the statistical analysis of the results was then presented with the aid of Statistica.

The next chapter will summarise the findings of the research study.

# Chapter 5: Conclusions

## 5.1 Introduction

The aim of this study is to compare the impact of data vault modelling and dimensional modelling on data warehouses in terms of performance and maintenance. This comparison includes data loading performance, query performance, data volumes and the flexibility of the particular model. This aim was addressed by means of four separate experiments:

1. *Load performance*. The objective of this experiment was to compare the historical and incremental data loads for each of the data models.
2. *Query performance*. The objective of this experiment was to compare the query performance for each of the data models. The query performance of the different categories of queries (a measure of complexity) was compared. There was also an overall query comparison between the two data models.
3. *Storage size*. The objective of this experiment was to compare the storage size of the two data models. This comparison was made in terms of data space, index space as well as total storage space.
4. *Flexibility*. The objective of this experiment was to determine the resilience of each the data models to changes or additions to business requirements.

This is the final chapter of the study and it will summarise and present the results of the above experiments in section 5.2, discuss any limitations to the research in section 5.3 and identify potential future areas of research on this topic in section 5.4.

## 5.2 Summary of experiment results

This section summarises and reviews the results that were obtained from each of the experiments.

### 5.2.1 Load performance

The research question that this particular experiment was required to answer involved how the two target models compared in terms of load performance. Load performance includes loading data from the source system into the two different data models for the initial/history load as well

as the incremental loads. Load performance was compared for historical and incremental data loads independently.

The hypotheses for this experiment were formulated as follows:

*The mean time required to load historical data into a data vault model is less than the mean time required to load the same data into a dimensional model.*

*The mean time required to load incremental data into a data vault model is less than the mean time required to load the same data into a dimensional model.*

The mean time to load historical data into the data vault was 225 seconds compared to 852 seconds for the dimensional model. Similarly, the mean time to load incremental data into the data vault was 155 seconds compared to 172 seconds for the dimensional model. Accordingly, the results from the experiment prove conclusively that both hypotheses may be accepted. Thus, the mean time required to load both historical and incremental data into a data vault model is less than the mean time required to load the same data into a dimensional model.

### **5.2.2 Query performance**

The research question that this particular experiment was required to answer involved how the two target models compared in terms of query performance.

The hypothesis for this experiment was formulated as follow:

*The mean query execution time for queries to a data vault model is less than the mean query execution time for similar queries to the dimensional Model.*

Different categories of queries were identified for the purposes of this experiment, with each category representing a set of typical data warehouse queries of varying complexity. A description of each query category is presented in table 5.1 below. Query performance for the two data models was compared for each of the categories as well as across all the categories.

**Table 5.1 Query categories in terms of dimensional models**

| Category | Description                                                                                                                                                          |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1        | Simple SELECT statement to a single dimension table. The only field allowed in a WHERE clause is a current indicator flag.                                           |
| 2        | Aggregated queries to a single dimension table that include WHERE, GROUP BY, HAVING and ORDER BY clauses.                                                            |
| 3        | Queries the relationship between two dimension tables through a single fact table. Queries include SELECT DISTINCT, WHERE and ORDER BY clauses.                      |
| 4        | Non-aggregated queries to a single fact table and all its related dimension tables. Queries include WHERE and ORDER BY clauses.                                      |
| 5        | Aggregated queries to a single fact table and some of its related dimension tables. Queries include aggregation functions, calculations, WHERE and GROUP BY clauses. |

The data analysis of the results revealed that the hypothesis can be rejected for all categories except categories two and three. The hypothesis is also rejected for queries combined across all categories. The analysis indicated that neither data vault nor dimensional modelling statistically performs better than the other as regards query performance.

### **5.2.3 Storage size**

This experiment was aimed at comparing the dimensional and data vault models in terms of the storage space required. Data and index storage space were considered independently and they were also combined.

The hypothesis for this experiment was formulated as follow:

*The mean storage space required for data in a data vault model is less than the mean storage space required for the same data in a dimensional model.*

The initial data analysis results summarised in table 4.24 appeared to indicate that the hypothesis could be accepted. The mean data storage space was 1713MB and 1727MB for the data vault and dimensional models respectively while the mean index storage space was 1850MB and 2071MB for the data vault and dimensional model respectively. The total mean storage space for the data vault model was 3564MB compared to 3799MB for the dimensional model.

However, a scatterplot of the storage space requirement prompted the researcher to analyse the results further. The results of the experiment were based on a set of data with 40 days of history. A time-series analysis was conducted to predict storage space requirements for more than 40 days of data. The results of this analysis indicated that the mean storage space

required for data in a data vault model would exceed that of the dimensional model after approximately 70 days of data was loaded into the data warehouse.

#### **5.2.4 Flexibility**

The aim of this experiment was to test the effect that changes in either existing requirements or additional requirements would have on each of the target data models.

The hypothesis for this experiment was formulated as follow:

*The number of existing queries that will be affected after changes/additions to the data vault model are made in order to meet changing user requirements is less than the number of existing queries that will be affected on a dimensional model after changes/additions to the model in order to meet the same set of changing requirements.*

Queries that either returned error messages or returned a different result set before and after implementing the changed requirements were classified as affected. Ideally, a warehouse should be able to adapt to changing requirements without affecting any existing queries.

The results from the experiment indicated that the hypothesis could be accepted. Out of a total of eleven queries, one query only was affected on the data vault model as compared to 4 queries on the dimensional model.

### **5.3 Limitations**

It is possible to generalise the results of this study. However, there are certain limitations to the study which may be categorised into one of the following two categories, namely, hardware and data.

The researcher was limited by the hardware available to him. The exact specification of the hardware used in the study is presented in Annexure O. This server was extremely limited when compared to data warehouse servers in the practice that typically have hundreds of gigabytes of RAM, 16 CPU cores and more and dedicated drives for the operation system files, SQL Server TempDB files, log files and data drives (Salch, Kraemer, Waheed & Dyke, 2009:21).

According to Linstedt, Graziano and Hultgren (2009b:27), the data vault model was designed with massively parallel processing (MPP) database technologies in mind and it would benefit

from MPP technologies more than would dimensional modelling. Examples of such database technologies include Microsoft's Parallel Data Warehouse, Terradata, and so on.

The TPC-E dataset that was used for this experiment also imposed certain limitations on the research results. There were two areas of concern with regards to the data that was used in the experiments; namely, quality and volume. As noted in chapter 4, section 4.4.2 there are no data quality issues with the TPC-E dataset. However, this is not typical of what happens in practice. According to Gartner Inc. (2005), "through 2007, more than 50 percent of data warehouse projects will have limited acceptance, or will be outright failures, as a result of a lack of attention to data quality issues". The researcher's concern is that poor quality data might have had an unexpected effect on the results.

The second concern regarding the dataset was that of the data volume. The total file size of the full set of TPC-E data that was used in this experiment was 4.9GB while the largest transactional set of data was the trade transactions 37 million records. However, this is small when compared to data warehouses that are hundreds of terabytes. Nevertheless, the size of the dataset was a function of the hardware that the researcher had available.

The impact of these limitations became the most obvious when the storage size experiment data was analysed. Recall from section 5.2.3 above that the results from the experiment seemed to indicate that the data vault model required less storage space than the dimensional model. However, upon further investigation using time-series predictive analysis, it emerged that the storage space required by the data vault model would exceed that of the dimensional model after approximately 70 days of history had been loaded. Thus, if the dataset that was used in the experiment had contained two years of data, the results would have been more accurate.

## **5.4 Future research**

Suggested future research on this topic is centred on the limitations which were discussed in section 5.3 above. The researcher recommends that a case study be conducted in order to implement a data vault model in an organisation that has an established data warehouse that is based on a dimensional model with hardware equivalent to that of the existing data warehouse. Implementing a data warehouse in a real organisation would make the results of this study more generalisable. The data would be in much larger volumes and there would be data quality issues.



Executing the case study at an organisation that made use of MPP technologies may also have an impact on the findings of this study and is worth investigating. This is as a result of the fact that MPP technologies would favour the data vault model more than the dimensional model (Linstedt *et al.*, 2009b:27).

In view of the fact that the experiments indicated that the models had different strengths and weaknesses, the researcher recommends that research be conducted that would consider the strengths and weaknesses of each technique and identify best use cases for each model in practice. The researcher was not able to find any previous research that indicated whether there is a correlation between the success of the data warehouse project and a specific data modelling technique. Such research would also yield useful results.

## **5.5 Chapter summary**

The aim of this research study was to scientifically compare data warehouse implementations based on dimensional modelling techniques with those based on data vault modelling techniques in terms of performance and maintenance through objective measurements.

Chapter 2 introduced various research methods that could be used to address the objective of the research study. Yin's process of elimination was used and a true experiment was selected as the appropriate research method (Yin, 2003:5). The selection of an experiment as the best possible research method was confirmed by Oates' definition of experiments (Oates, 2006:127).

The aim of chapter 3 was to introduce the reader to the two data warehouse modelling techniques that were relevant to this study; namely, dimensional modelling, a technique pioneered by Ralph Kimball (Kimball *et al.*, 2008:1), and data vault modelling, pioneered by Dan Linstedt (Linstedt, 2002). The chapter concluded with a theoretical comparison between dimensional modelling and data vault modelling.

The execution and results of the experiment were reported in chapter 4. The environment and details of each of the experiments was discussed. For each experiment, the researcher formulated hypotheses and predicted the outcomes of the experiments. Finally, the results were presented and discussed.

Even though the results were conclusive in all of the experiments with the exception of the query performance experiment, neither model performed best in all the experiments. In terms of

load performance and flexibility the data vault model outperformed the dimensional model while the dimensional model outperformed the data vault model in the storage and query performance experiments. However, different results may have been obtained if a significantly larger bigger dataset with real-life characteristics, such as data quality problems, had been used and also if MPP database technologies had been used. These were identified as limitations of the research. Future research that would address these limitations was discussed.

Based on the research results and the analysis of the experiments the data vault model may be said to be a viable alternative to dimensional modelling.



## References

- Adelman, S. & Moss, L. 2000. Data warehouse failures. <http://www.tdan.com/view-articles/4876> Date of access: 6 Aug. 2012.
- Al-Hakim, L. & Carter-Steel, A. 2009. Information systems research methods, epistemology, and applications. Hershey, PA: IGI Global. 395 p.
- Ali, S., Rauf, A. & Javed, H. 2010. A technique for handling range and fuzzy match queries on encrypted data. *The International Arab Journal of Information Technology*, 10(3). <http://www.ccis2k.org/iajit/PDF/vol.10,no.3/8-3022.pdf> Date of access: 10 Nov. 2012.
- Anon. 2005. Largest commercial database in Winter Corp: top ten survey tops one hundred terabytes. [http://www.wintercorp.com/PressReleases/ttp2005\\_pressrelease\\_091405.htm](http://www.wintercorp.com/PressReleases/ttp2005_pressrelease_091405.htm) Date of access: 12 Jan. 2012.
- Anon. 2010. TPC Benchmark E: standard specification Version 1.12.0. <http://www.tpc.org/tpce/spec/v1.12.0/TPCE-v1.12.0.pdf> Date of access: 12 Jul. 2011.
- Anon. 2011a. TPC Benchmarks. <http://www.tpc.org/information/benchmarks.asp> Date of access: 12 Jul. 2011.
- Anon. 2011b. What is the TPC? <http://www.tpc.org/information/about/about.asp> Date of access: 12 Jul. 2011.
- Bakerville, R. 1999. Investigating information systems with action research. *Communications of Association of Information Systems*, 2(19) Nov.:1–32.
- Baum, D. 2001. Interview: Gartner Group's Howard Dresner. *Information Builders Magazine*, 11(2) Aug.:26–28.
- Becker, B. 2006. Kimball University's design tip #81: fact table surrogate keys. <http://www.ralphkimball.com/html/designtipsPDF/DesignTips2006/KU81FactTableSurrogateKeys.pdf> Date of access: 14 Apr. 2012.
- Bernard, H.R. 2000. Social research methods: qualitative and quantitative approaches. Thousand Oaks, Calif.: Sage. 784p.

- Chua, W.F. 1986. Radical developments in accounting thought. *The Accounting Review*, 61(4) Oct.:601–632.
- Curiel, E. 2001. Against the excesses of quantum gravity: a plea for modesty. *Philosophy of Science*, 68(3) Sept.:424–441.
- Curino, A., Jones, E., Zhang, Y. & Madden, S. 2010. Schism: a workload-driven approach to database replication and partitioning. *Proceedings of the VLDS Endowment*, 3(1):48–57.
- De Villiers, M.R. 2005. Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory. *Proceedings of the 2005 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, Vol. 150:142–151.
- Dodig-Crnkovic, G. 2002. Scientific methods in computer science. *Conference for the Promotion of Research in IT, Skövde, April 2002*. <http://www.mrtc.mdh.se/publications/0446.pdf> Date of access: 30 Aug. 2012.
- Drewek, K. 2005. Data warehouse architecture: the great debate. <http://www.b-eye-network.com/view/693> Date of access: 3 Mar. 2012.
- Eckerson, W. 2005. Four ways to build a data warehouse. <http://tdwi.org/Articles/2003/05/12/Four-Ways-to-Build-a-Data-Warehouse.aspx?Page=1> Date of access: 3 Mar. 2012.
- Evers, M. 2011. (dm\_unseen@hotmail.com) Dan Linstedt forums: initial indexing strategy for data vault. E-mail to: Van Schalkwyk, M. (mariusvanschalkwyk@gmail.com) 15 Jan. 2011.
- Feinberg, D. & Beyer, M.A. 2011. Magic quadrant for data warehouse database management systems, Gartner. <http://www.gartner.com/technology/media-products/reprints/teradata/vol3/article1/article1.html> Date of access: 5 Oct. 2011.
- Gartner Inc. 2005. Gartner says more than 50 percent of data warehouse projects will have limited acceptance or will be failures through 2007. <http://www.gartner.com/it/page.jsp?id=492112> Data of access: 5 Dec. 2012.

Gartner Inc. 2012. Magic quadrant for business intelligence platforms. <http://www.gartner.com/technology/reprints.do?id=1-196WFCB&ct=120207&st=sb> Date of access: 6 Dec. 2012.

Goede, R. 2005. A framework for the explicit use of specific systems thinking methodologies in data-driven decision support system development. DPhil thesis, University of Pretoria, Pretoria. 273p.

Hirschheim, R.A. 1985. Information systems epistemology: an historical perspective. <http://areadocenti.eco.unicas.it/virili/TerracinaRW/Kit/HirschheimISEpistemology.pdf> Date of access: 2 Sep. 2012.

Hirschheim, R.A. & Klein, H.K. 1989. Four paradigms of information systems development. *Communications of the ACM*, 32(10) Oct.:1199–1216.

Hirschheim, R.A. & Klein, H.K. 1994. Realising emancipatory principals in information systems development: the case for ETHICS. *MIS Quarterly*, 18(1) Mar.:83–109.

Inmon, W.H. 2002. Building the data warehouse. 3rd ed. New York: Wiley. 412p.

Inmon, W.H., Strauss, D. & Neushloss, G. 2008. DW2.0: the architecture for the next generation of data warehousing. New York: Elsevier. 371p.

Kimball, R. 2001. Kimball design tip #29: graceful modifications to existing fact and dimension tables. <http://www.rkimball.com/html/designtipsPDF/DesignTips2001/KimballDT29Graceful.pdf> Date of access: 28 Sep. 2012.

Kimball, R. & Caserta, J. 2004. The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming and delivering data. New York: Wiley. 491p.

Kimball, R., Reeves, R., Ross, M. & Thornthwaite, W. 1998. The data warehouse lifecycle toolkit: expert methods for designing, developing, and deploying data warehouses. New York: Wiley. 771p.

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J. & Becker, B. 2008. The data warehouse lifecycle toolkit: practical techniques for building data warehouse and business intelligence systems. 2nd ed. New York: Wiley. 635p.

Klein, H.K. & Myers, M.D. 1999. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, 23(1) Mar.:67–94.

Linstedt, D.E. 2002. Data vault series 1: data vault overview. <http://www.tdan.com/view-articles/5054/> Date of access: 22 Aug. 2012.

Linstedt, D.E. 2003. Data vault series 2: data vault components. <http://www.tdan.com/view-articles/5155/> Date of access: 29 Oct. 2012.

Linstedt, D.E. 2004. Data vault series 4: link tables. <http://www.tdan.com/view-articles/5172> Date of access: 2 Aug. 2012.

Linstedt, D.E. 2010a. (danlistedt@gmail.com) Question on the satellite of transactional links. E-mail to: Van Schalkwyk, M. (mariusvanschalkwyk@gmail.com) 28 Jul 2010.

Linstedt, D.E. 2010b. Data vault has many joins: part 1. <http://danlinstedt.com/2010/03/data-vault-has-many-joins-part-1/> Date of access: 2 Aug. 2012.

Linstedt, D.E. 2010c. Lesson 1: cross-reference and lookups. <http://danlinstedt.com/my-coach/data-vault-structure-lessons/structure-cross-references/lesson-1-cross-reference-and-lookups/> Date of access: 30 Jul. 2012.

Linstedt, D.E. 2010d. Lesson 2: modelling lookups in data vault. <http://danlinstedt.com/my-coach/data-vault-structure-lessons/structure-cross-references/lesson-2-modeling-lookups-in-data-vault/> Date of access: 30 Jul. 2010.

Linstedt, D.E. 2010e. Lesson 4: handling code collisions. <http://danlinstedt.com/my-coach/data-vault-structure-lessons/structure-cross-references/lesson-4-handling-code-collisions/> Date of access: 22 Jul. 2012.

Linstedt, D.E. 2010f. Sequence numbers: for EDW & BI projects. <http://danlinstedt.com/2010/06/sequence-numbers-for-edw-bi-projects/comment-page-1/#comment-91> Date of access: 2 Aug. 2012.

Linstedt, D.E. & Graziano, K. 2010. How to build an effective data vault model. [http://danlinstedt.com/wp-content/uploads/2010/06/Data-Vault-Modeling\\_v6.3.pdf](http://danlinstedt.com/wp-content/uploads/2010/06/Data-Vault-Modeling_v6.3.pdf) Date of access: 10 Sep. 2012.

Linstedt, D. & Graziano, K. 2011. Super charge your data warehouse: invaluable data modelling rules to implement your data vault. 152p.

Linstedt, D.E., Graziano, K. & Hultgren, H. 2009a. The business of data vault modelling. 2nd ed. Lulu. 98p.

Linstedt, D.E., Graziano, K. & Hultgren, H. 2009b. The new business supermodel: the business of data vault data modelling. 2nd ed. Lulu. 81p.

Microsoft. 2011. Microsoft developer network: optimising data warehouse query performance through bitmap filtering. [http://msdn.microsoft.com/en-us/library/bb522541\(v=SQL.100\).aspx](http://msdn.microsoft.com/en-us/library/bb522541(v=SQL.100).aspx) Date of access: 4 Jan. 2012.

Microsoft. 2012. Estimating the size of a nonclustered index. [http://msdn.microsoft.com/en-us/library/ms190620\(v=sql.105\).aspx](http://msdn.microsoft.com/en-us/library/ms190620(v=sql.105).aspx) Date of access: 30 Nov. 2012.

Mingers, J. 2001. Combining IS research methods: towards a pluralist methodology. *Information Systems Research*, 12(3) Sep.:240–259.

Mundy, J., Thornthwaite, W. & Kimball, R. 2011. The Microsoft data warehouse toolkit: with SQL Server 2008 R2 and the business intelligence toolset. 2nd ed. Indiana.: Wiley. 648p.

Myers, M.D. 2010. Qualitative research in information systems. <http://www.qual.auckland.ac.nz/> Date of access: 1 Sep. 2012.

Oates, B.J. 2006. Researching information systems and computing. Thousand Oaks, Calif.: Sage. 341p.

Olivier, M.S. 2004. Information technology research: a practical guide for computer science and informatics. 2nd ed. Pretoria: Van Schaik. 147p.

Orlikowski, W.J. & Baroudi, J.J. 1991. Studying information technology in organizations: research approaches and assumptions. *Information Systems Research*, 2(1) Mar.:1–28.

Perkins, A. 2000. Critical success factors for data warehouse engineering part 1. <http://www.tdan.com/view-articles/4836> Date of access: 6 Aug. 2012.



Petty, C. 2007. Gartner says world wide database market increased 14 percent in 2006. <http://www.gartner.com/it/page.jsp?id=507466> Date of access: 5 Oct. 2011.

Raisinghani, M.S. 2000. Adapting data modelling techniques for data warehouse design. *The Journal of Computer Information Systems*, 40(3) Apr:73–77.

Salch, D., Kraemer, E., Waheed, U. & Dyke, P. 2009. Fast track data warehouse 3.0 reference guide. Microsoft. p47.

Steyn, A.G.W, Smit, C.F., Du Toit, S.H.C. & Strasheim, C. 1998. Moderne statistiek vir die praktyk. 6th ed. Pretoria: Van Schaik. 761p.

Straub, D., Gefen, D. & Boudreau, M. 2005. The ISWorld quantitative, positivist research methods website. <http://www.dstraub.cis.gsu.edu:88/quant/> Date of access: 13 Sep. 2012.

Thomas, R. 1996. Research methods: guidance for postgraduates. 2nd ed. Greenfield, T. New York: Oxford University Press. 306p.

Trochim, W.M. 2012. The research methods knowledge base. 2nd ed. <http://www.socialresearchmethods.net/kb> Date of access: 12 Sep. 2012.

Walsham, G. 1995. The emergence of interpretivism in IS research. *Information Systems Research*, 6(4) Dec.:176–394.

Yin, R.K. 2003. Case study research: design and methods. 3rd ed. Thousand Oaks, Calif.: Sage. 181p.

Yin, R.K. & Heald, K.A. 1975. Using the case survey method to analyse policy studies. *Administrative Science Quarterly*, 20(3) Sep.:371–381.

## **Annexures**

All the annexures referenced throughout the research are contained on CD at the back of this report. Each annexure is a separate Adobe pdf document (.pdf).

## Appendix A - Disabled Windows Services

Several Microsoft Windows services were disabled to prevent these processes from unpredictably consuming memory or processor time during the execution of an experiment. *Table 1* below lists the each of the services on the server, its status, start-up type and under which account the service is running.

**Table 1 Windows Services**

| Name                                    | Description                                                                                                                                                                                                                                                                                                                                      | Status  | Start-up Type | Log On As     |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------|---------------|
| Application Experience                  | Processes application compatibility cache requests for applications as they are launched.                                                                                                                                                                                                                                                        | Started | Manual        | Local System  |
| Application Identity                    | Determines and verifies the identity of an application. Disabling this service will prevent AppLocker from being enforced.                                                                                                                                                                                                                       |         | Manual        | Local Service |
| Application Information                 | Facilitates the running of interactive applications with additional administrative privileges. If this service is stopped users will be unable to launch applications with the additional administrative privileges they may require to perform desired user tasks.                                                                              |         | Manual        | Local System  |
| Application Layer Gateway Service       | Provides support for 3rd party protocol plug-ins for Internet Connection Sharing.                                                                                                                                                                                                                                                                |         | Disabled      | Local Service |
| Application Management                  | Processes installation removal and enumeration requests for software deployed through Group Policy. If the service is disabled users will be unable to install remove or enumerate software deployed through Group Policy. If this service is disabled any services that explicitly depend on it will fail to start.                             |         | Disabled      | Local System  |
| Background Intelligent Transfer Service | Transfers files in the background using idle network bandwidth. If the service is disabled then any applications that depend on BITS such as Windows Update or MSN Explorer will be unable to automatically download programs and other information.                                                                                             |         | Disabled      | Local System  |
| Base Filtering Engine                   | The Base Filtering Engine (BFE) is a service that manages firewall and Internet Protocol security (IPsec) policies and implements user mode filtering. Stopping or disabling the BFE service will significantly reduce the security of the system. It will also result in unpredictable behaviour in IPsec management and firewall applications. |         | Disabled      | Local Service |

| Name                                   | Description                                                                                                                                                                                                                                                                                                                                                   | Status  | Start-up Type | Log On As     |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------|---------------|
| Certificate Propagation                | Copies user certificates and root certificates from smart cards into the current user's certificate store detects when a smart card is inserted into a smart card reader and if needed installs the smart card Plug and Play mini driver.                                                                                                                     |         | Disabled      | Local System  |
| CNG Key Isolation                      | The CNG key isolation service is hosted in the LSA process. The service provides key process isolation to private keys and associated cryptographic operations as required by the Common Criteria. The service stores and uses long-lived keys in a secure process complying with Common Criteria requirements.                                               |         | Manual        | Local System  |
| COM+ Event System                      | Supports System Event Notification Service (SENS) which provides automatic distribution of events to subscribing Component Object Model (COM) components. If the service is stopped SENS will close and will not be able to provide logon and logoff notifications. If this service is disabled any services that explicitly depend on it will fail to start. | Started | Automatic     | Local Service |
| COM+ System Application                | Manages the configuration and tracking of Component Object Model (COM)+-based components. If the service is stopped most COM+-based components will not function properly. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                          |         | Manual        | Local System  |
| Computer Browser                       | Maintains an updated list of computers on the network and supplies this list to computers designated as browsers. If this service is stopped this list will not be updated or maintained. If this service is disabled any services that explicitly depend on it will fail to start.                                                                           |         | Disabled      | Local System  |
| Credential Manager                     | Provides secure storage and retrieval of credentials to users' applications and security service packages.                                                                                                                                                                                                                                                    |         | Disabled      | Local System  |
| Cryptographic Services                 | Provides four management services: Catalogue Database Service which confirms the signatures of Windows files and allows new programs to be installed.                                                                                                                                                                                                         |         |               |               |
| DCOM Server Process Launcher           | The DCOMLAUNCH service launches COM and DCOM servers in response to object activation requests. If this service is stopped or disabled programs using COM or DCOM will not function properly. It is strongly recommended that you have the DCOMLAUNCH service running.                                                                                        | Started | Automatic     | Local System  |
| Desktop Window Manager Session Manager | Provides Desktop Window Manager start-up and maintenance services.                                                                                                                                                                                                                                                                                            |         | Disabled      | Local System  |
| DHCP Client                            | Registers and updates IP addresses and DNS records for this computer. If this service is stopped this computer will not receive dynamic IP addresses and DNS updates. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                               |         | Automatic     | Local Service |

| Name                                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Status  | Start-up Type             | Log On As       |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------------------|-----------------|
| Diagnostic Policy Service           | The Diagnostic Policy Service enables problem detection troubleshooting and resolution for Windows components. If this service is stopped diagnostics will no longer function.                                                                                                                                                                                                                                                                                         |         | Disabled                  | Local Service   |
| Diagnostic Service Host             | The Diagnostic Service Host is used by the Diagnostic Policy Service to host diagnostics that need to run in a Local Service context. If this service is stopped any diagnostics that depend on it will no longer function.                                                                                                                                                                                                                                            |         | Disabled                  | Local Service   |
| Diagnostic System Host              | The Diagnostic System Host is used by the Diagnostic Policy Service to host diagnostics that need to run in a Local System context. If this service is stopped any diagnostics that depend on it will no longer function.                                                                                                                                                                                                                                              |         | Disabled                  | Local System    |
| Disk Defragmenter                   | Provides Disk Defragmentation Capabilities.                                                                                                                                                                                                                                                                                                                                                                                                                            |         | Manual                    | Local System    |
| Distributed Link Tracking Client    | Maintains links between NTFS files within a computer or across computers in a network.                                                                                                                                                                                                                                                                                                                                                                                 |         | Disabled                  | Local System    |
| Distributed Transaction Coordinator | Coordinates transactions that span multiple resource managers such as databases message queues and file systems. If this service is stopped these transactions will fail. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                    | Started | Automatic (Delayed Start) | Network Service |
| DNS Client                          | The DNS Client service (dnscache) caches Domain Name System (DNS) names and registers the full computer name for this computer. If the service is stopped DNS names will continue to be resolved. However the results of DNS name queries will not be cached and the computer's name will not be registered. If the service is disabled any services that explicitly depend on it will fail to start.                                                                  |         | Disabled                  | Network Service |
| Encrypting File System (EFS)        | Provides the core file encryption technology used to store encrypted files on NTFS file system volumes. If this service is stopped or disabled applications will be unable to access encrypted files.                                                                                                                                                                                                                                                                  |         | Disabled                  | Local System    |
| Extensible Authentication Protocol  | The Extensible Authentication Protocol (EAP) service provides network authentication in such scenarios as 802.1x wired and wireless VPN and Network Access Protection (NAP). EAP also provides application programming interfaces (APIs) that are used by network access clients including wireless and VPN clients during the authentication process. If you disable this service this computer is prevented from accessing networks that require EAP authentication. |         | Disabled                  | Local System    |

| Name                                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Status | Start-up Type | Log On As     |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|---------------|---------------|
| Function Discovery Provider Host        | The FDPHOST service hosts the Function Discovery (FD) network discovery providers. These FD providers supply network discovery services for the Simple Services Discovery Protocol (SSDP) and Web Services – Discovery (WS-D) protocol. Stopping or disabling the FDPHOST service will disable network discovery for these protocols when using FD. When this service is unavailable network services using FD and relying on these discovery protocols will be unable to find network devices or resources. |        | Disabled      | Local Service |
| Function Discovery Resource Publication | Publishes this computer and resources attached to this computer so they can be discovered over the network. If this service is stopped network resources will no longer be published and they will not be discovered by other computers on the network.                                                                                                                                                                                                                                                      |        | Disabled      | Local Service |
| Group Policy Client                     | The service is responsible for applying settings configured by administrators for the computer and users through the Group Policy component. If the service is stopped or disabled the settings will not be applied and applications and components will not be manageable through Group Policy. Any components or applications that depend on the Group Policy component might not be functional if the service is stopped or disabled.                                                                     |        | Automatic     | Local System  |
| Health Key and Certificate Management   | Provides X.509 certificate and key management services for the Network Access Protection Agent (NAPAgent). Enforcement technologies that use X.509 certificates may not function properly without this service.                                                                                                                                                                                                                                                                                              |        | Disabled      | Local System  |
| Human Interface Device Access           | Enables generic input access to Human Interface Devices (HID) which activates and maintains the use of predefined hot buttons on keyboards remote controls and other multimedia devices. If this service is stopped hot buttons controlled by this service will no longer function. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                |        | Disabled      | Local System  |
| IKE and AuthIP IPsec Keying Modules     | The IKEEXT service hosts the Internet Key Exchange (IKE) and Authenticated Internet Protocol (AuthIP) keying modules. These keying modules are used for authentication and key exchange in Internet Protocol security (IPsec). Stopping or disabling the IKEEXT service will disable IKE and AuthIP key exchange with peer computers. IPsec is typically configured to use IKE or AuthIP.                                                                                                                    |        |               |               |

| Name                                                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Status | Start-up Type | Log On As       |
|-------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|---------------|-----------------|
| Interactive Services Detection                        | Enables user notification of user input for interactive services which enables access to dialogs created by interactive services when they appear. If this service is stopped notifications of new interactive service dialogs will no longer function and there might not be access to interactive service dialogs. If this service is disabled both notifications of and access to new interactive service dialogs will no longer function.                                                                                       |        | Disabled      | Local System    |
| Internet Connection Sharing (ICS)                     | Provides network address translation addressing name resolution and/or intrusion prevention services for a home or small office network.                                                                                                                                                                                                                                                                                                                                                                                            |        | Disabled      | Local System    |
| IP Helper                                             | Provides tunnel connectivity using IPv6 transition technologies (6to4ISATAPPort Proxy and Teredo)and IP-HTTPS. If this service is stopped the computer will not have the enhanced connectivity benefits that these technologies offer.                                                                                                                                                                                                                                                                                              |        | Disabled      | Local System    |
| IPsec Policy Agent                                    | Internet Protocol security (IPsec) supports network-level peer authentication data origin authentication data integrity data confidentiality (encryption)and replay protection. This service enforces IPsec policies created through the IP Security Policies snap-in or the command-line tool "netsh ipsec". If you stop this service you may experience network connectivity issues if your policy requires that connections use IPsec. Also remote management of Windows Firewall is not available when this service is stopped. |        | Disabled      | Network Service |
| KtmRm for Distributed Transaction Coordinator         | Coordinates transactions between the Distributed Transaction Coordinator (MSDTC) and the Kernel Transaction Manager (KTM). If it is not needed it is recommended that this service remain stopped. If it is needed both MSDTC and KTM will start this service automatically. If this service is disabled any MSDTC transaction interacting with a Kernel Resource Manager will fail and any services that explicitly depend on it will fail to start.                                                                               |        | Manual        | Network Service |
| Link-Layer Topology Discovery Mapper                  | Creates a Network Map consisting of PC and device topology (connectivity) information and metadata describing each PC and device. If this service is disabled the Network Map will not function properly.                                                                                                                                                                                                                                                                                                                           |        | Disabled      | Local Service   |
| Microsoft .NET Framework NGEN v2.0.50727_X64          | Microsoft .NET Framework NGEN.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |        | Manual        | Local System    |
| Microsoft .NET Framework NGEN v2.0.50727_X86          | Microsoft .NET Framework NGEN.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |        | Manual        | Local System    |
| Microsoft Fibre Channel Platform Registration Service | Registers the platform with all available Fibre Channel fabrics and maintains the registrations.                                                                                                                                                                                                                                                                                                                                                                                                                                    |        | Manual        | Local Service   |

| Name                                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Status  | Start-up Type | Log On As       |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------|-----------------|
| Microsoft iSCSI Initiator Service       | Manages Internet SCSI (iSCSI) sessions from this computer to remote iSCSI target devices. If this service is stopped this computer will not be able to login or access iSCSI targets. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                                                                                                                                         |         | Disabled      | Local System    |
| Microsoft Software Shadow Copy Provider | Manages software-based volume shadow copies taken by the Volume Shadow Copy service. If this service is stopped software-based volume shadow copies cannot be managed. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                                                                                                                                                        |         | Manual        | Local System    |
| Multimedia Class Scheduler              | Enables relative prioritization of work based on system-wide task priorities. This is intended mainly for multimedia applications. If this service is stopped individual tasks resort to their default priority.                                                                                                                                                                                                                                                                                                                                                                                        |         | Manual        | Local System    |
| Net.Tcp Port Sharing Service            | Provides ability to share TCP ports over the net.tcp protocol.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |         | Disabled      | Local Service   |
| Netlogon                                | Maintains a secure channel between this computer and the domain controller for authenticating users and services. If this service is stopped the computer may not authenticate users and services and the domain controller cannot register DNS records. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                                                                      |         | Disabled      | Local System    |
| Network Access Protection Agent         | The Network Access Protection (NAP) agent service collects and manages health information for client computers on a network. Information collected by NAP agent is used to make sure that the client computer has the required software and settings. If a client computer is not compliant with health policy it can be provided with restricted network access until its configuration is updated. Depending on the configuration of health policy client computers might be automatically updated so that users quickly regain full network access without having to manually update their computer. |         | Disabled      | Network Service |
| Network Connections                     | Manages objects in the Network and Dial-Up Connections folder in which you can view both local area network and remote connections.                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |         | Manual        | Local System    |
| Network List Service                    | Identifies the networks to which the computer has connected collects and stores properties for these networks and notifies applications when these properties change.                                                                                                                                                                                                                                                                                                                                                                                                                                   | Started | Manual        | Local Service   |
| Network Location Awareness              | Collects and stores configuration information for the network and notifies programs when this information is modified. If this service is stopped configuration information might be unavailable. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                                                                                                                             | Started | Automatic     | Network Service |



| Name                                                | Description                                                                                                                                                                                                                                                                                                                                             | Status  | Start-up Type | Log On As       |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------|-----------------|
| Network Store Interface Service                     | This service delivers network notifications (e.g. interface addition/deleting etc.) to user mode clients. Stopping this service will cause loss of network connectivity. If this service is disabled any other services that explicitly depend on this service will fail to start.                                                                      | Started | Automatic     | Local Service   |
| Office Source Engine                                | Saves installation files used for updates and repairs and is required for the downloading of Setup updates and Watson error reports.                                                                                                                                                                                                                    |         | Manual        | Local System    |
| Office Software Protection Platform                 | Office Software Protection Platform Service (unlocalized description).                                                                                                                                                                                                                                                                                  |         | Manual        | Network Service |
| Performance Counter DLL Host                        | Enables remote users and 64-bit processes to query performance counters provided by 32-bit DLLs. If this service is stopped only local users and 32-bit processes will be able to query performance counters provided by 32-bit DLLs.                                                                                                                   |         | Manual        | Local Service   |
| Performance Logs & Alerts                           | Performance Logs and Alerts Collects performance data from local or remote computers based on preconfigured schedule parameters then writes the data to a log or triggers an alert. If this service is stopped performance information will not be collected. If this service is disabled any services that explicitly depend on it will fail to start. |         | Manual        | Local Service   |
| Plug and Play                                       | Enables a computer to recognize and adapt to hardware changes with little or no user input. Stopping or disabling this service will result in system instability.                                                                                                                                                                                       |         | Automatic     | Local System    |
| PnP-X IP Bus Enumerator                             | The PnP-X bus enumerator service manages the virtual network bus. It discovers network connected devices using the SSDP/WS discovery protocols and gives them presence in PnP. If this service is stopped or disabled presence of NCD devices will not be maintained in PnP. All pnpX based scenarios will stop functioning.                            |         | Disabled      | Local System    |
| Portable Device Enumerator Service                  | Enforces group policy for removable mass-storage devices. Enables applications such as Windows Media Player and Image Import Wizard to transfer and synchronize content using removable mass-storage devices.                                                                                                                                           |         | Disabled      | Local System    |
| Power                                               | Manages power policy and power policy notification delivery.                                                                                                                                                                                                                                                                                            |         | Automatic     | Local System    |
| Print Spooler                                       | Loads files to memory for later printing.                                                                                                                                                                                                                                                                                                               |         | Automatic     | Local System    |
| Problem Reports and Solutions Control Panel Support | This service provides support for viewing sending and deletion of system-level problem reports for the Problem Reports and Solutions control panel.                                                                                                                                                                                                     |         | Disabled      | Local System    |
| Protected Storage                                   | Provides protected storage for sensitive data such as passwords to prevent access by unauthorized services processes or users.                                                                                                                                                                                                                          |         | Manual        | Local System    |

| Name                                             | Description                                                                                                                                                                                                                                                                                                                                                                   | Status  | Start-up Type | Log On As       |
|--------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------|-----------------|
| Remote Access Auto Connection Manager            | Creates a connection to a remote network whenever a program references a remote DNS or NetBIOS name or address.                                                                                                                                                                                                                                                               |         | Manual        | Local System    |
| Remote Access Connection Manager                 | Manages dial-up and virtual private network (VPN) connections from this computer to the Internet or other remote networks. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                          | Started | Manual        | Local System    |
| Remote Desktop Configuration                     | Remote Desktop Configuration service (RDCS) is responsible for all Remote Desktop Services and Remote Desktop related configuration and session maintenance activities that require SYSTEM context. These include per-session temporary folders RD themes and RD certificates.                                                                                                |         | Disabled      | Local System    |
| Remote Desktop Services                          | Allows users to connect interactively to a remote computer. Remote Desktop and Remote Desktop Session Host Server depend on this service. To prevent remote use of this computer clear the checkboxes on the Remote tab of the System properties control panel item.                                                                                                          |         | Disabled      | Network Service |
| Remote Desktop Services UserMode Port Redirector | Allows the redirection of Printers/Drives/Ports for RDP connections.                                                                                                                                                                                                                                                                                                          |         | Disabled      | Local System    |
| Remote Procedure Call (RPC)                      | The RPCSS service is the Service Control Manager for COM and DCOM servers. It performs object activations requests object exporter resolutions and distributed garbage collection for COM and DCOM servers. If this service is stopped or disabled programs using COM or DCOM will not function properly. It is strongly recommended that you have the RPCSS service running. | Started | Automatic     | Network Service |
| Remote Procedure Call (RPC) Locator              | In Windows 2003 and earlier versions of Windows the Remote Procedure Call (RPC) Locator service manages the RPC name service database. In Windows Vista and later versions of Windows this service does not provide any functionality and is present for application compatibility.                                                                                           |         | Disabled      | Network Service |
| Remote Registry                                  | Enables remote users to modify registry settings on this computer. If this service is stopped the registry can be modified only by users on this computer. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                          |         | Disabled      | Local Service   |
| Resultant Set of Policy Provider                 | Provides a network service that processes requests to simulate application of Group Policy settings for a target user or computer in various situations and computes the Resultant Set of Policy settings.                                                                                                                                                                    |         | Manual        | Local System    |
| Routing and Remote Access                        | Offers routing services to businesses in local area and wide area network environments.                                                                                                                                                                                                                                                                                       |         | Disabled      | Local System    |

| Name                                      | Description                                                                                                                                                                                                                                                                                                                                                                         | Status  | Start-up Type                | Log On As       |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|------------------------------|-----------------|
| RPC Endpoint Mapper                       | Resolves RPC interfaces identifiers to transport endpoints. If this service is stopped or disabled programs using Remote Procedure Call (RPC) services will not function properly.                                                                                                                                                                                                  | Started | Automatic                    | Network Service |
| Secondary Logon                           | Enables starting processes under alternate credentials. If this service is stopped this type of logon access will be unavailable. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                         |         | Manual                       | Local System    |
| Secure Socket Tunnelling Protocol Service | Provides support for the Secure Socket Tunnelling Protocol (SSTP) to connect to remote computers using VPN. If this service is disabled users will not be able to use SSTP to access remote servers.                                                                                                                                                                                | Started | Manual                       | Local Service   |
| Security Accounts Manager                 | The start-up of this service signals other services that the Security Accounts Manager (SAM) is ready to accept requests. Disabling this service will prevent other services in the system from being notified when the SAM is ready which may in turn cause those services to fail to start correctly. This service should not be disabled.                                        | Started | Automatic                    | Local System    |
| Server                                    | Supports file print and named-pipe sharing over the network for this computer. If this service is stopped these functions will be unavailable. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                            | Started | Automatic                    | Local System    |
| Shell Hardware Detection                  | Provides notifications for AutoPlay hardware events.                                                                                                                                                                                                                                                                                                                                |         | Automatic                    | Local System    |
| Smart Card                                | Manages access to smart cards read by this computer. If this service is stopped this computer will be unable to read smart cards. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                         |         | Disabled                     | Local Service   |
| Smart Card Removal Policy                 | Allows the system to be configured to lock the user desktop upon smart card removal.                                                                                                                                                                                                                                                                                                |         | Disabled                     | Local System    |
| SNMP Trap                                 | Receives trap messages generated by local or remote Simple Network Management Protocol (SNMP) agents and forwards the messages to SNMP management programs running on this computer. If this service is stopped SNMP-based programs on this computer will not receive SNMP trap messages. If this service is disabled any services that explicitly depend on it will fail to start. |         | Disabled                     | Local Service   |
| Software Protection                       | Enables the download installation and enforcement of digital licenses for Windows and Windows applications. If the service is disabled the operating system and licensed applications may run in a notification mode. It is strongly recommended that you not disable the Software Protection service.                                                                              | Started | Automatic<br>(Delayed Start) | Network Service |
| Special Administration Console Helper     | Allows administrators to remotely access a command prompt using Emergency Management Services.                                                                                                                                                                                                                                                                                      |         | Manual                       | Local System    |

| Name                                | Description                                                                                                                                                                                                                                                                                                                                             | Status  | Start-up Type | Log On As       |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------|-----------------|
| SPP Notification Service            | Provides Software Licensing activation and notification.                                                                                                                                                                                                                                                                                                |         | Manual        | Local Service   |
| SQL Active Directory Helper Service | Enables integration with Active Directories.                                                                                                                                                                                                                                                                                                            |         | Disabled      | Network Service |
| SQL Server (MSSQLSERVER)            | Provides storage processing and controlled access of data and rapid transaction processing.                                                                                                                                                                                                                                                             | Started | Automatic     | Local System    |
| SQL Server Agent (MSSQLSERVER)      | Executes jobs monitors SQL Server fires alerts and allows automation of some administrative tasks.                                                                                                                                                                                                                                                      |         | Disabled      | Local System    |
| SQL Server Browser                  | Provides SQL Server connection information to client computers.                                                                                                                                                                                                                                                                                         |         | Disabled      | Local Service   |
| SQL Server VSS Writer               | Provides the interface to backup/restore Microsoft SQL server through the Windows VSS infrastructure.                                                                                                                                                                                                                                                   |         | Automatic     | Local System    |
| SSDP Discovery                      | Discovers networked devices and services that use the SSDP discovery protocol such as UPnP devices. Also announces SSDP devices and services running on the local computer. If this service is stopped SSDP-based devices will not be discovered. If this service is disabled any services that explicitly depend on it will fail to start.             |         | Disabled      | Local Service   |
| System Event Notification Service   | Monitors system events and notifies subscribers to COM+ Event System of these events.                                                                                                                                                                                                                                                                   |         | Automatic     | Local System    |
| Task Scheduler                      | Enables a user to configure and schedule automated tasks on this computer. The service also hosts multiple Windows system-critical tasks. If this service is stopped or disabled these tasks will not be run at their scheduled times. If this service is disabled any services that explicitly depend on it will fail to start.                        | Started | Automatic     | Local System    |
| TCP/IP NetBIOS Helper               | Provides support for the NetBIOS over TCP/IP (NetBT) service and NetBIOS name resolution for clients on the network therefore enabling users to share files print and log on to the network. If this service is stopped these functions might be unavailable. If this service is disabled any services that explicitly depend on it will fail to start. | Started | Automatic     | Local Service   |
| Telephony                           | Provides Telephony API (TAPI) support for programs that control telephony devices on the local computer and through the LAN on servers that are also running the service.                                                                                                                                                                               | Started | Manual        | Network Service |
| Thread Ordering Server              | Provides ordered execution for a group of threads within a specific period of time.                                                                                                                                                                                                                                                                     |         | Manual        | Local Service   |
| TPM Base Services                   | Enables access to the Trusted Platform Module (TPM) which provides hardware-based cryptographic services to system components and applications. If this service is stopped or disabled applications will be unable to use keys protected by the TPM.                                                                                                    |         | Disabled      | Local Service   |

| Name                                                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Status  | Start-up Type | Log On As     |
|--------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------|---------------|
| UPnP Device Host                                       | Allows UPnP devices to be hosted on this computer. If this service is stopped any hosted UPnP devices will stop functioning and no additional hosted devices can be added. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                                                                        |         | Disabled      | Local Service |
| User Profile Service                                   | This service is responsible for loading and unloading user profiles. If this service is stopped or disabled users will no longer be able to successfully logon or logoff applications may have problems getting to users' data and components registered to receive profile event notifications will not receive them.                                                                                                                                                                                                      | Started | Automatic     | Local System  |
| Virtual Disk                                           | Provides management services for disks volumes file systems and storage arrays.                                                                                                                                                                                                                                                                                                                                                                                                                                             |         | Manual        | Local System  |
| Volume Shadow Copy                                     | Manages and implements Volume Shadow Copies used for backup and other purposes. If this service is stopped shadow copies will be unavailable for backup and the backup may fail. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                                                                  |         | Manual        | Local System  |
| Windows Audio                                          | Manages audio for Windows-based programs. If this service is stopped audio devices and effects will not function properly. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                                                                                                                        |         | Manual        | Local Service |
| Windows Audio Endpoint Builder                         | Manages audio devices for the Windows Audio service. If this service is stopped audio devices and effects will not function properly. If this service is disabled any services that explicitly depend on it will fail to start                                                                                                                                                                                                                                                                                              |         | Manual        | Local System  |
| Windows CardSpace                                      | Securely enables the creation management and disclosure of digital identities.                                                                                                                                                                                                                                                                                                                                                                                                                                              |         | Manual        | Local System  |
| Windows Colour System                                  | The WcsPlugInService service hosts third-party Windows Colour System colour device model and gamut map model plug-in modules. These plug-in modules are vendor-specific extensions to the Windows Colour System baseline colour device and gamut map models. Stopping or disabling the WcsPlugInService service will disable this extensibility feature and the Windows Colour System will use its baseline model processing rather than the vendor's desired processing. This might result in inaccurate colour rendering. |         | Disabled      | Local Service |
| Windows Driver Foundation - User-mode Driver Framework | Manages user-mode driver host processes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |         | Manual        | Local System  |
| Windows Error Reporting Service                        | Allows errors to be reported when programs stop working or responding and allows existing solutions to be delivered. Also allows logs to be generated for diagnostic and repair services. If this service is stopped error reporting might not work correctly and results of diagnostic services and repairs might not be displayed.                                                                                                                                                                                        |         | Disabled      | Local System  |

| Name                                               | Description                                                                                                                                                                                                                                                                                                                                                                      | Status  | Start-up Type             | Log On As       |
|----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------------------|-----------------|
| Windows Event Collector                            | This service manages persistent subscriptions to events from remote sources that support WS-Management protocol. This includes Windows Vista event logs hardware and IPMI-enabled event sources. The service stores forwarded events in a local Event Log. If this service is stopped or disabled event subscriptions cannot be created and forwarded events cannot be accepted. |         | Manual                    | Network Service |
| Windows Event Log                                  | This service manages events and event logs. It supports logging events querying events subscribing to events archiving event logs and managing event metadata. It can display events in both XML and plain text format. Stopping this service may compromise security and reliability of the system.                                                                             | Started | Automatic                 | Local Service   |
| Windows Firewall                                   | Windows Firewall helps protect your computer by preventing unauthorized users from gaining access to your computer through the Internet or a network.                                                                                                                                                                                                                            |         | Disabled                  | Local Service   |
| Windows Font Cache Service                         | Optimizes performance of applications by caching commonly used font data. Applications will start this service if it is not already running. It can be disabled though doing so will degrade application performance.                                                                                                                                                            | Started | Automatic (Delayed Start) | Local Service   |
| Windows Installer                                  | Adds modifies and removes applications provided as a Windows Installer (*.msi) package. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                |         | Manual                    | Local System    |
| Windows Management Instrumentation                 | Provides a common interface and object model to access management information about operating system devices applications and services. If this service is stopped most Windows-based software will not function properly. If this service is disabled any services that explicitly depend on it will fail to start.                                                             | Started | Automatic                 | Local System    |
| Windows Modules Installer                          | Enables installation modification and removal of Windows updates and optional components. If this service is disabled install or uninstall of Windows updates might fail for this computer.                                                                                                                                                                                      |         | Manual                    | Local System    |
| Windows Presentation Foundation Font Cache 3.0.0.0 | Optimizes performance of Windows Presentation Foundation (WPF) applications by caching commonly used font data. WPF applications will start this service if it is not already running. It can be disabled though doing so will degrade the performance of WPF applications.                                                                                                      |         | Manual                    | Local Service   |

| Name                                      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Status  | Start-up Type             | Log On As       |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------------------|-----------------|
| Windows Remote Management (WS-Management) | Windows Remote Management (WinRM) service implements the WS-Management protocol for remote management. WS-Management is a standard web services protocol used for remote software and hardware management. The WinRM service listens on the network for WS-Management requests and processes them. The WinRM Service needs to be configured with a listener using winrm.cmd command line tool or through Group Policy in order for it to listen over the network. The WinRM service provides access to WMI data and enables event collection. Event collection and subscription to events require that the service is running. WinRM messages use HTTP and HTTPS as transports. The WinRM service does not depend on IIS but is preconfigured to share a port with IIS on the same machine. The WinRM service reserves the /wsman URL prefix. To prevent conflicts with IIS administrators should ensure that any websites hosted on IIS do not use the /wsman URL prefix. |         | Disabled                  | Network Service |
| Windows Time                              | Maintains date and time synchronization on all clients and servers in the network. If this service is stopped date and time synchronization will be unavailable. If this service is disabled any services that explicitly depend on it will fail to start.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |         | Manual                    | Local Service   |
| Windows Update                            | Enables the detection download and installation of updates for Windows and other programs. If this service is disabled users of this computer will not be able to use Windows Update or its automatic updating feature and programs will not be able to use the Windows Update Agent (WUA) API.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Started | Automatic (Delayed Start) | Local System    |
| WinHTTP Web Proxy Auto-Discovery Service  | WinHTTP implements the client HTTP stack and provides developers with a Win32 API and COM Automation component for sending HTTP requests and receiving responses. In addition WinHTTP provides support for auto-discovering a proxy configuration via its implementation of the Web Proxy Auto-Discovery (WPAD) protocol.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |         | Disabled                  | Local Service   |
| Wired AutoConfig                          | The Wired AutoConfig (DOT3SVC) service is responsible for performing IEEE 802.1X authentication on Ethernet interfaces. If your current wired network deployment enforces 802.1X authentication the DOT3SVC service should be configured to run for establishing Layer 2 connectivity and/or providing access to network resources. Wired networks that do not enforce 802.1X authentication are unaffected by the DOT3SVC service.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |         | Disabled                  | Local System    |
| WMI Performance Adapter                   | Provides performance library information from Windows Management Instrumentation (WMI) providers to clients on the network. This service only runs when Performance Data Helper is activated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |         | Manual                    | Local System    |

| Name        | Description                                                                                                                                                                                                                                            | Status  | Start-up Type | Log On As       |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------|-----------------|
| Workstation | Creates and maintains client network connections to remote servers using the SMB protocol. If this service is stopped these connections will be unavailable. If this service is disabled any services that explicitly depend on it will fail to start. | Started | Automatic     | Network Service |



## Appendix B - Sample Data Generation

EGenLoader is an application that is provided as part of the TPC-E v1.12.0 specification (Anon, 2010:204). The application is used to generate the sample TPC-E data in text files which is used for the benchmark.

EGenLoader was used to generate forty days of TPC-E data using the following command and parameters:

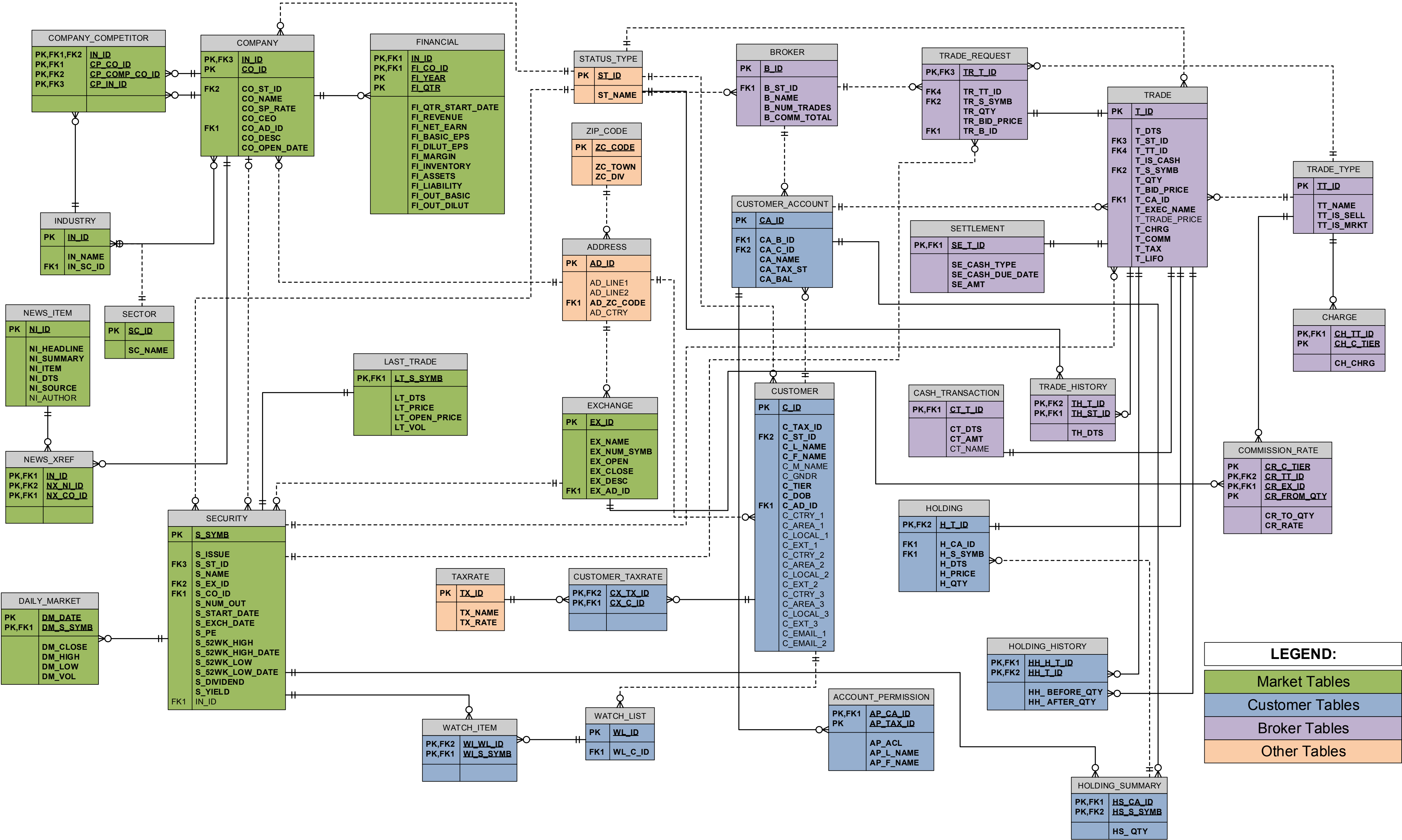
```
EGenLoader -f 500 -c 55000 -w 40
```

This command generates a TPC-E dataset with a scale factor of 500 (-f 500), 55 000 customers (-c 55000) and forty working days of transactions (-w 40). The first ten days were used as history data. The remaining thirty days were used for thirty incremental loads, one day each.

The history as well as incremental data files is included on the appendix CD.

Appendix C - Full Sized TPC-E ERD

The insert below is a larger version of the TPC-E ERD as shown in chapter 4. The ERD is derived from the TPC benchmark E specification document (Anon, 2010) which documents the tables and relationships between the tables.



## Appendix D - TPC-E Data Dictionary

This appendix contains a detailed data dictionary of the TPC-E data model that was derived from the TPC-E version 1.12.0 benchmark (Anon, 2010). The data types have been changed to reflect Microsoft SQL Server 2008 data types.

### **Table:** ACCOUNT\_PERMISSION

**Description:** This table contains information about the access the customer or an individual other than the customer has to a given customer account. Customer accounts may have trades executed on them by more than one person.

**Table 1 TPC-E ACCOUNT\_PERMISSION table description**

| Field Name | Data Type   | Constraints | Description                                                                          |
|------------|-------------|-------------|--------------------------------------------------------------------------------------|
| AP_CA_ID   | BIGINT      | Not Null    | Customer account identifier.                                                         |
| AP_ACL     | VARCHAR(4)  | Not Null    | Access control list defining the permissions the person has on the customer account. |
| AP_TAX_ID  | VARCHAR(20) | Not Null    | Tax identifier of the person with access to the customer account.                    |
| AP_L_NAME  | VARCHAR(25) | Not Null    | Last name of the person with access to the customer account.                         |
| AP_F_NAME  | VARCHAR(20) | Not Null    | First name of the person with access to the customer account.                        |

### **Table:** ADDRESS

**Description:** This table contains address information for companies, exchanges and customers. Each company, exchange and customer has one address record.

**Table 2 TPC-E ADDRESS table description**

| Field Name | Data Type   | Constraints | Description         |
|------------|-------------|-------------|---------------------|
| AD_ID      | BIGINT      | Not Null    | Address identifier. |
| AD_LINE1   | VARCHAR(80) |             | Address Line 1.     |
| AD_LINE2   | VARCHAR(80) |             | Address Line 2.     |
| AD_ZC_CODE | VARCHAR(12) | Not Null    | Zip or postal code. |
| AD_CTRY    | VARCHAR(80) |             | Country.            |

**Table:** BROKER

**Description:** The table contains information about brokers. There is one record for each broker.

**Table 3 TPC-E BROKER table description**

| Field Name   | Data Type      | Constraints | Description                                                                |
|--------------|----------------|-------------|----------------------------------------------------------------------------|
| B_ID         | BIGINT         | Not Null    | Broker identifier.                                                         |
| B_ST_ID      | VARCHAR(4)     | Not Null    | Broker status type identifier. Identifies if this broker is active or not. |
| B_NAME       | VARCHAR(49)    | Not Null    | Broker's name.                                                             |
| B_NUM_TRADES | INT            | Not Null    | Number of trades this broker has executed so far.                          |
| B_COMM_TOTAL | DECIMAL(12, 2) | Not Null    | Amount of commission this broker has earned so far.                        |

**Table:** CASH\_TRANSACTION

**Description:** The table contains information about cash transactions.

**Table 4 TPC-E CASH\_TRANSACTION table description**

| Field Name | Data Type      | Constraints | Description                                                                                          |
|------------|----------------|-------------|------------------------------------------------------------------------------------------------------|
| CT_T_ID    | BIGINT         | Not Null    | Trade identifier.                                                                                    |
| CT_DTS     | DATETIME2(0)   | Not Null    | Date and time stamp of when the transaction took place.                                              |
| CT_AMT     | DECIMAL(10, 2) | Not Null    | Amount of the cash transaction.                                                                      |
| CT_NAME    | VARCHAR(100)   | Not Null    | Transaction name, or description: e.g. "Buy Keebler Cookies", "Cash from sale of DuPont stock", etc. |

**Table:** CHARGE

**Description:** The table contains information about charges for placing a trade request. Charges are based on the customer's tier and the trade type.

**Table 5 TPC-E CHARGE table description**

| Field Name | Data Type      | Constraints          | Description                         |
|------------|----------------|----------------------|-------------------------------------|
| CH_TT_ID   | VARCHAR(3)     | Not Null             | Trade type identifier.              |
| CH_C_TIER  | TINYINT        | Not Null<br>In 1,2,3 | Customer's tier.                    |
| CH_CHRG    | DECIMAL(10, 2) | Not Null<br>>= 0     | Charge for placing a trade request. |

**Table: COMMISSION\_RATE**

**Description:** The commission rate depends on several factors: the tier the customer is in, the type of trade, the quantity of securities traded, and the exchange that executes the trade.

**Table 6 TPC-E COMMISSION\_RATE table description**

| Field Name  | Data Type     | Constraints                  | Description                                                         |
|-------------|---------------|------------------------------|---------------------------------------------------------------------|
| CR_C_TIER   | TINYINT       | Not Null<br>in 1,2,3         | Customer's tier. Valid values 1, 2 or 3.                            |
| CR_TT_ID    | VARCHAR(3)    | Not Null                     | Trade Type identifier. Identifies the type of trade.                |
| CR_EX_ID    | VARCHAR(6)    | Not Null                     | Exchange identifier. Identifies the exchange the trade is against.  |
| CR_FROM_QTY | INT           | Not Null<br>>= 0             | Lower bound of quantity being traded to match this commission rate. |
| CR_TO_QTY   | INT           | Not Null<br>><br>CR_FROM_QTY | Upper bound of quantity being traded to match this commission rate. |
| CR_RATE     | DECIMAL(5, 2) | Not Null<br>>= 0             | Commission rate. Ranges from 0.00 to 100.00. Example: 10% is 10.00. |

**Table: COMPANY**

**Description:** The table contains information about all companies with publicly traded securities. There is one record for each company.

**Table 7 TPC-E COMPANY table description**

| Field Name   | Data Type    | Constraints | Description                                                                  |
|--------------|--------------|-------------|------------------------------------------------------------------------------|
| CO_ID        | BIGINT       | Not Null    | Company identifier.                                                          |
| CO_ST_ID     | VARCHAR(4)   | Not Null    | Company status type identifier. Identifies if this company is active or not. |
| CO_NAME      | VARCHAR(60)  | Not Null    | Company name.                                                                |
| CO_IN_ID     | VARCHAR(2)   | Not Null    | Industry identifier of the industry the company is in.                       |
| CO_SP_RATE   | VARCHAR(4)   | Not Null    | Company's credit rating from Standard & Poor.                                |
| CO_CEO       | VARCHAR(46)  | Not Null    | Name of company's Chief Executive Officer.                                   |
| CO_AD_IN     | BIGINT       | Not Null    | Address identifier.                                                          |
| CO_DESC      | VARCHAR(150) | Not Null    | Company description.                                                         |
| CO_OPEN_DATE | DATE         | Not Null    | Date the company was founded.                                                |

**Table:** COMPANY\_COMPETITOR

**Description:** This table contains information for the competitors of a given company and the industry in which the company competes.

**Table 8 TPC-E COMPANY\_COMPETITOR table description**

| Field Name    | Data Type  | Constraints | Description                                                                                                                                                                     |
|---------------|------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CP_CO_ID      | BIGINT     | Not Null    | Company identifier.                                                                                                                                                             |
| CP_COMP_CO_ID | BIGINT     | Not Null    | Company identifier of the competitor company for the specified industry.                                                                                                        |
| CP_IN_ID      | VARCHAR(2) | Not Null    | Industry identifier of the industry in which the CP_CO_ID company considers that the CP_COMP_CO_ID company competes with it. This may not be either company's primary industry. |

**Table:** CUSTOMER

**Description:** This table contains information about the customers of the brokerage firm. There is one row for each customer.

**Table 9 TPC-E CUSTOMER table description**

| Field Name | Data Type   | Constraints          | Description                                                                                                                                 |
|------------|-------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| C_ID       | BIGINT      | Not Null             | Customer identifier, used internally to link customer information.                                                                          |
| C_TAX_ID   | VARCHAR(20) | Not Null             | Customer's tax identifier, used externally on communication to the customer. Is alphanumeric.                                               |
| C_ST_ID    | VARCHAR(4)  | Not Null             | Customer status type identifier. Identifies if this customer is active or not.                                                              |
| C_L_NAME   | VARCHAR(25) | Not Null             | Primary Customer's last name.                                                                                                               |
| C_F_NAME   | VARCHAR(20) | Not Null             | Primary Customer's first name.                                                                                                              |
| C_M_NAME   | VARCHAR(1)  |                      | Primary Customer's middle name initial.                                                                                                     |
| C_GNDR     | VARCHAR(1)  |                      | Gender of the primary customer. Valid values are "M" for male or "F" for female.                                                            |
| C_TIER     | TINYINT     | Not Null<br>in 1,2,3 | Customer tier: tier 1 accounts are charged highest fees, tier 2 accounts are charged medium fees, and tier 3 accounts have the lowest fees. |
| C_DOB      | DATE        | Not Null             | Customer's date of birth.                                                                                                                   |
| C_AD_ID    | BIGINT      | Not Null             | Address identifier of the customer's address.                                                                                               |
| C_CTRY_1   | VARCHAR(3)  |                      | Country code for customer's phone 1.                                                                                                        |
| C_AREA_1   | VARCHAR(3)  |                      | Area code for customer's phone 1.                                                                                                           |
| C_LOCAL_1  | VARCHAR(10) |                      | Local number for customer's phone 1.                                                                                                        |

| Field Name | Data Type   | Constraints | Description                              |
|------------|-------------|-------------|------------------------------------------|
| C_EXT_1    | VARCHAR(5)  |             | Extension number for customer's phone 1. |
| C_CTRY_2   | VARCHAR(3)  |             | Country code for customer's phone 2.     |
| C_AREA_2   | VARCHAR(3)  |             | Area code for customer's phone 2.        |
| C_LOCAL_2  | VARCHAR(10) |             | Local number for customer's phone 2.     |
| C_EXT_2    | VARCHAR(5)  |             | Extension number for customer's phone 2. |
| C_CTRY_3   | VARCHAR(3)  |             | Country code for customer's phone 3.     |
| C_AREA_3   | VARCHAR(3)  |             | Area code for customer's phone 3.        |
| C_LOCAL_3  | VARCHAR(10) |             | Local number for customer's phone 3.     |
| C_EXT_3    | VARCHAR(5)  |             | Extension number for customer's phone 3. |
| C_EMAIL_1  | VARCHAR(50) |             | Customer's e-mail address 1.             |
| C_EMAIL_2  | VARCHAR(50) |             | Customer's e-mail address 2.             |

**Table:** CUSTOMER\_ACCOUNT

**Description:** The table contains account information related to accounts of each customer. Each customer has at least one account, but can have more than one account.

**Table 10 TPC-E CUSTOMER\_ACCOUNT table description**

| Field Name | Data Type     | Constraints          | Description                                                                                                                                                                                      |
|------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CA_ID      | BIGINT        | Not Null             | Customer account identifier.                                                                                                                                                                     |
| CA_B_ID    | BIGINT        | Not Null             | Broker identifier of the broker who manages this customer account.                                                                                                                               |
| CA_C_ID    | BIGINT        | Not Null             | Customer identifier of the customer who owns this account.                                                                                                                                       |
| CA_NAME    | VARCHAR(50)   | Not Null             | Name of customer account. Example, "Trish Hogan 401(k)".                                                                                                                                         |
| CA_TAX_ST  | TINYINT       | Not Null<br>in 0,1,2 | Tax status of this account: 0 means this account is not taxable, 1 means this account is taxable and tax must be withheld, 2 means this account is taxable and tax does not have to be withheld. |
| CA_BAL     | DECIMAL(12,2) | Not Null             | Account's cash balance.                                                                                                                                                                          |

**Table: CUSTOMER\_TAXRATE**

**Description:** The table contains two references per customer into the TAXRATE table. One reference is for state/province tax; the other one is for national tax. The table contains the actual tax rates.

**Table 11 TPC-E CUSTOMER\_TAXRATE table description**

| Field Name | Data Type  | Constraints | Description                                                    |
|------------|------------|-------------|----------------------------------------------------------------|
| CX_TX_ID   | VARCHAR(4) | Not Null    | Tax rate identifier.                                           |
| CX_C_ID    | BIGINT     | Not Null    | Customer identifier of a customer that must pay this tax rate. |

**Table: DAILY\_MARKET**

**Description:** The table contains daily market statistics for each security, using the closing market data from the last completed trading day. There is one record per security per day.

**Table 12 TPC-E DAILY\_MARKET table description**

| Field Name | Data Type     | Constraints | Description                         |
|------------|---------------|-------------|-------------------------------------|
| DM_DATE    | DATE          | Not Null    | Date of last completed trading day. |
| DM_S_SYMB  | VARCHAR(15)   | Not Null    | Security symbol of this security.   |
| DM_CLOSE   | DECIMAL(8, 2) | Not Null    | Closing price for this security.    |
| DM_HIGH    | DECIMAL(8, 2) | Not Null    | Day's High price for this security. |
| DM_LOW     | DECIMAL(8, 2) | Not Null    | Day's Low price for this security.  |
| DM_VOL     | BIGINT        | Not Null    | Day's volume for this security.     |

**Table: EXCHANGE**

**Description:** The table contains information about financial exchanges. There is one record for each exchange.

**Table 13 TPC-E EXCHANGE table description**

| Field Name  | Data Type    | Constraints | Description                                                           |
|-------------|--------------|-------------|-----------------------------------------------------------------------|
| EX_ID       | VARCHAR(6)   | Not Null    | Exchange identifier. Values are, "NYSE", "NASDAQ", "AMEX",   "PCX"  . |
| EX_NAME     | VARCHAR(100) | Not Null    | Exchange name.                                                        |
| EX_NUM_SYMB | INT          | Not Null    | Number of securities traded on this exchange.                         |
| EX_OPEN     | SMALLINT     | Not Null    | Exchange daily start time expressed in GMT.                           |
| EX_CLOSE    | SMALLINT     | Not Null    | Exchange daily stop time, expressed in GMT.                           |



| Field Name | Data Type    | Constraints | Description                  |
|------------|--------------|-------------|------------------------------|
| EX_DESC    | VARCHAR(150) | Not Null    | Description of the exchange. |
| EX_AD_ID   | BIGINT       | Not Null    | Mailing address of exchange. |

**Table:** FINANCIAL

**Description:** The table contains information about a company's quarterly financial reports. There is one record per company per quarter. This is fiscal quarters.

**Table 14 TPC-E FINANCIAL table description**

| Field Name        | Data Type      | Constraints            | Description                                                                    |
|-------------------|----------------|------------------------|--------------------------------------------------------------------------------|
| FI_CO_ID          | BIGINT         | Not Null               | Company identifier.                                                            |
| FI_YEAR           | SMALLINT       | Not Null               | Year of the quarter end.                                                       |
| FI_QTR            | TINYINT        | Not Null<br>in 1,2,3,4 | Quarter number that the financial information is for: valid values 1, 2, 3, 4. |
| FI_QTR_START_DATE | DATE           | Not Null               | Start date of quarter.                                                         |
| FI_REVENUE        | DECIMAL(15, 2) | Not Null               | Reported revenue for the quarter.                                              |
| FI_NET_EARN       | DECIMAL(15, 2) | Not Null               | Net earnings reported for the quarter.                                         |
| FI_BASIC_EPS      | DECIMAL(10, 2) | Not Null               | Basic earnings per share reported for the quarter.                             |
| FI_DILUT_EPS      | DECIMAL(10, 2) | Not Null               | Diluted earnings per share reported for the quarter.                           |
| FI_MARGIN         | DECIMAL(10, 2) | Not Null               | Profit divided by revenues for the quarter.                                    |
| FI_INVENTORY      | DECIMAL(15, 2) | Not Null               | Value of inventory on hand at the end of the quarter.                          |
| FI_ASSETS         | DECIMAL(15, 2) | Not Null               | Value of total assets at the end of the quarter.                               |
| FI_LIABILITY      | DECIMAL(15, 2) | Not Null               | Value of total liabilities at the end of the quarter.                          |
| FI_OUT_BASIC      | BIGINT         | Not Null               | Average number of common shares outstanding (basic).                           |
| FI_OUT_DILUT      | BIGINT         | Not Null               | Average number of common shares outstanding (diluted).                         |

**Table: HOLDING**

**Description:** The table contains information about the customer account's security holdings.

**Table 15 TPC-E HOLDING table description**

| Field Name | Data Type     | Constraints     | Description                               |
|------------|---------------|-----------------|-------------------------------------------|
| H_T_ID     | BIGINT        | Not Null        | Trade Identifier of the trade.            |
| H_CA_ID    | BIGINT        | Not Null        | Customer account identifier.              |
| H_S_SYMB   | VARCHAR(15)   | Not Null        | Symbol for the security held.             |
| H_DTS      | DATETIME2(0)  | Not Null        | Date this security was purchased or sold. |
| H_PRICE    | DECIMAL(8, 2) | Not Null<br>> 0 | Unit purchase price of this security.     |
| H_QTY      | INT           | Not Null        | Quantity of this security held.           |

**Table: HOLDING\_HISTORY**

**Description:** The table contains information about holding positions that were inserted, updated or deleted and which trades made each change.

**Table 16 TPC-E HOLDING\_HISTORY table description**

| Field Name      | Data Type | Constraints | Description                                                                                                                                                                             |
|-----------------|-----------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HH_H_T_ID       | BIGINT    | Not Null    | Trade Identifier of the trade that originally created the holding row. This is a Foreign Key to the TRADE table rather than the HOLDING table because the HOLDING row could be deleted. |
| HH_T_ID         | BIGINT    | Not Null    | Trade Identifier of the current trade (the one that last inserted, updated or deleted the holding identified by HH_H_T_ID).                                                             |
| [HH_BEFORE_QTY] | INT       | Not Null    | Quantity of this security held before the modifying trade. On initial insertion, HH_BEFORE_QTY is 0.                                                                                    |
| [HH_AFTER_QTY]  | INT       | Not Null    | Quantity of this security held after the modifying trade. If the HOLDING row gets deleted by the modifying trade, then HH_AFTER_QTY is 0.                                               |

**Table:** HOLDING\_SUMMARY

**Description:** The table contains aggregate information about the customer account's security holdings.

**Table 17 TPC-E HOLDING\_SUMMARY table description**

| Field Name | Data Type   | Constraints | Description                           |
|------------|-------------|-------------|---------------------------------------|
| HS_CA_ID   | BIGINT      | Not Null    | Customer account identifier.          |
| HS_S_SYMB  | VARCHAR(15) | Not Null    | Symbol for the security held.         |
| [HS_QTY]   | INT         | Not Null    | Total quantity of this security held. |

**Table:** INDUSTRY

**Description:** The table contains information about the industries that a company can belong to. There is one record per industry. Each company belongs to only one industry.

**Table 18 TPC-E INDUSTRY table description**

| Field Name | Data Type   | Constraints | Description                                                                                                  |
|------------|-------------|-------------|--------------------------------------------------------------------------------------------------------------|
| IN_ID      | VARCHAR(2)  | Not Null    | Industry identifier.                                                                                         |
| IN_NAME    | VARCHAR(50) | Not Null    | Industry name. Examples: "Air Travel", "Air Cargo", "Software", "Consumer Banking", "Merchant Banking", etc. |
| IN_SC_ID   | VARCHAR(2)  | Not Null    | Sector identifier of the sector the industry is in.                                                          |

**Table:** LAST\_TRADE

**Description:** The table contains one row for each security with the latest trade price and volume for each security.

**Table 19 TPC-E LAST\_TRADE table description**

| Field Name    | Data Type     | Constraints | Description                                                                             |
|---------------|---------------|-------------|-----------------------------------------------------------------------------------------|
| LT_S_SYMB     | VARCHAR(15)   | Not Null    | Security symbol.                                                                        |
| LT_DTS        | DATETIME2(0)  | Not Null    | Date and timestamp of when this row was last updated.                                   |
| LT_PRICE      | DECIMAL(8, 2) | Not Null    | Latest trade price for this security.                                                   |
| LT_OPEN_PRICE | DECIMAL(8, 2) | Not Null    | Price the security opened at today.                                                     |
| LT_VOL        | BIGINT        | Not Null    | Volume of trading on the market for this security so far today. Value initialized to 0. |

**Table:** NEWS\_ITEM

**Description:** The table contains information about news items of interest.

**Table 20 TPC-E NEWS\_ITEM table description**

| Field Name  | Data Type    | Constraints | Description                                                                    |
|-------------|--------------|-------------|--------------------------------------------------------------------------------|
| NI_ID       | BIGINT       | Not Null    | News item identifier.                                                          |
| NI_HEADLINE | VARCHAR(80)  | Not Null    | News item headline.                                                            |
| NI_SUMMARY  | VARCHAR(255) | Not Null    | News item summary.                                                             |
| NI_ITEM     | VARCHAR(MAX) | Not Null    | Large object containing the news item or links to the story.                   |
| NI_DTS      | DATETIME2(0) | Not Null    | Date and time the news item was published.                                     |
| NI_SOURCE   | VARCHAR(30)  | Not Null    | Source of the news item.                                                       |
| NI_AUTHOR   | VARCHAR(30)  | Not Null    | Author of the news item. May be NULL if the news item came off a wire service. |

**Table:** NEWS\_XREF

**Description:** The table contains a cross-reference of news items to companies that are mentioned in the news item.

**Table 21 TPC-E NEWS\_XREF table description**

| Field Name | Data Type | Constraints | Description                                                                             |
|------------|-----------|-------------|-----------------------------------------------------------------------------------------|
| NX_NI_ID   | BIGINT    | Not Null    | News item identifier.                                                                   |
| NX_CO_ID   | BIGINT    | Not Null    | Company identifier of the company (or one of the companies) mentioned in the news item. |

**Table:** SECTOR

**Description:** The table contains information about market sectors that an industry. There is one record per sector. Each industry belongs to only one sector.

**Table 22 TPC-E SECTOR table description**

| Field Name | Data Type   | Constraints | Description                                                                      |
|------------|-------------|-------------|----------------------------------------------------------------------------------|
| SC_ID      | VARCHAR(2)  | Not Null    | Sector identifier.                                                               |
| SC_NAME    | VARCHAR(30) | Not Null    | Sector name. Examples: "Energy", "Materials", "Industrials", "Health Care", etc. |

**Table: SECURITY**

**Description:** This table contains information about each security traded on any of the exchanges. A security is traded at only one exchange. There is one record for each security.

**Table 23 TPC-E SECURITY table description**

| Field Name       | Data Type      | Constraints | Description                                                                    |
|------------------|----------------|-------------|--------------------------------------------------------------------------------|
| S_SYMB           | VARCHAR(15)    | Not Null    | Security symbol used to identify the security on "ticker".                     |
| S_ISSUE          | VARCHAR(6)     | Not Null    | Security issue type. Example: "COMMON", "PERF_A", "PERF_B", etc.               |
| S_ST_ID          | VARCHAR(4)     | Not Null    | Security status type identifier. Identifies if this security is active or not. |
| S_NAME           | VARCHAR(70)    | Not Null    | Security name.                                                                 |
| S_EX_ID          | VARCHAR(6)     | Not Null    | Exchange identifier of the exchange the security is traded on.                 |
| S_CO_ID          | BIGINT         | Not Null    | Company identifier of the company this security is issued by.                  |
| S_NUM_OUT        | BIGINT         | Not Null    | Number of shares outstanding for this security.                                |
| S_START_DATE     | DATE           | Not Null    | Date security first started trading.                                           |
| S_EXCH_DATE      | DATE           | Not Null    | Date security first started trading on this exchange.                          |
| S_PE             | DECIMAL(10, 2) | Not Null    | Current share price to earnings per share ratio.                               |
| S_52WK_HIGH      | DECIMAL(8, 2)  | Not Null    | Security share price 52-week high.                                             |
| S_52WK_HIGH_DATE | DATE           | Not Null    | Date of security share price 52-week high.                                     |
| S_52WK_LOW       | DECIMAL(8, 2)  | Not Null    | Security share price 52-week low.                                              |
| S_52WK_LOW_DATE  | DATE           | Not Null    | Date of security share price 52-week low.                                      |
| S_DIVIDEND       | DECIMAL(10, 2) | Not Null    | Annual Dividend per share amount. May be zero, is not allowed to be negative.  |
| S_YIELD          | DECIMAL(5, 2)  | Not Null    | Dividend to share price ratio. Value is in percent. Example 10.00 is 10%       |

**Table: SETTLEMENT**

**Description:** The table contains information about how trades are settled: specifically whether the settlement is on margin or in cash and when the settlement is due.

**Table 24 TPC-E SETTLEMENT table description**

| Field Name       | Data Type      | Constraints | Description                                                                             |
|------------------|----------------|-------------|-----------------------------------------------------------------------------------------|
| SE_T_ID          | BIGINT         | Not Null    | Trade identifier.                                                                       |
| SE_CASH_TYPE     | VARCHAR(40)    | Not Null    | Type of cash settlement involved: possible values "Margin", "Cash Account".             |
| SE_CASH_DUE_DATE | DATE           | Not Null    | Date by which customer or brokerage must receive the cash; date of trade plus two days. |
| SE_AMT           | DECIMAL(10, 2) | Not Null    | Cash amount of settlement.                                                              |

**Table: STATUS\_TYPE**

**Description:** This table holds status codes and descriptions for brokers, companies, customers, securities, and trade transactions. There is one record for each distinct status.

**Table 25 TPC-E STATUS\_TYPE table description**

| Field Name | Data Type   | Constraints | Description                                                                            |
|------------|-------------|-------------|----------------------------------------------------------------------------------------|
| ST_ID      | VARCHAR(4)  | Not Null    | Status type identifier.                                                                |
| ST_NAME    | VARCHAR(10) | Not Null    | Status value. Examples: "Active", "Completed", "Pending", "Cancelled" and "Submitted". |

**Table: TAXRATE**

**Description:** The table contains information about tax rates.

**Table 26 TPC-E TAXRATE table description**

| Field Name | Data Type     | Constraints     | Description                                                                             |
|------------|---------------|-----------------|-----------------------------------------------------------------------------------------|
| TX_ID      | VARCHAR(4)    | Not Null        | Tax rate identifier. Format - two letters followed by one digit. Examples: "US1", "CA1" |
| TX_NAME    | VARCHAR(50)   | Not Null        | Tax rate name.                                                                          |
| TX_RATE    | DECIMAL(6, 5) | Not Null<br>> 0 | Tax rate, between 0.00000 and 1.00000, inclusive.                                       |

**Table: TRADE**

**Description:** The table contains information about trades that brokers do on behalf of customers. There is one record for every trade transaction that takes place.

**Table 27 TPC-E TRADE table description**

| Field Name    | Data Type      | Constraints        | Description                                                                                                                                                                                      |
|---------------|----------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T_ID          | BIGINT         | Not Null           | Trade identifier.                                                                                                                                                                                |
| T_DTS         | DATETIME2(0)   | Not Null           | Date and time of trade.                                                                                                                                                                          |
| T_ST_ID       | VARCHAR(4)     | Not Null           | Status type identifier; identifies the status of this trade.                                                                                                                                     |
| T_TT_ID       | VARCHAR(3)     | Not Null           | Trade type identifier; identifies the type of his trade.                                                                                                                                         |
| T_IS_CASH     | BIT            | Not Null<br>in 0,1 | Is this trade a cash (1) or margin (0) trade?                                                                                                                                                    |
| T_S_SYMB      | VARCHAR(15)    | Not Null           | Security symbol of the security that was traded.                                                                                                                                                 |
| T_QTY         | INT            | Not Null<br>> 0    | Quantity of securities traded.                                                                                                                                                                   |
| T_BID_PRICE   | DECIMAL(8, 2)  | Not Null<br>> 0    | The requested unit price.                                                                                                                                                                        |
| T_CA_ID       | BIGINT         | Not Null           | Customer account identifier.                                                                                                                                                                     |
| T_EXEC_NAME   | VARCHAR(49)    | Not Null           | Name of the person executing the trade.                                                                                                                                                          |
| T_TRADE_PRICE | DECIMAL(8, 2)  | Not Null           | Unit price at which the security was traded.                                                                                                                                                     |
| T_CHRG        | DECIMAL(10, 2) | Not Null<br>≥ 0    | Fee charged for placing this trade request.                                                                                                                                                      |
| T_COMM        | DECIMAL(10, 2) | Not Null<br>≥ 0    | Commission earned on this trade; may be zero.                                                                                                                                                    |
| T_TAX         | DECIMAL(10, 2) | Not Null<br>≥ 0    | Amount of tax due on this trade; can be zero. Whether the tax is withheld from the settlement amount depends on the customer account tax status.                                                 |
| T_LIFO        | BIT            | Not Null<br>in 0,1 | If this trade is closing an existing position, is it executed against the newest-to-oldest account holdings of this security (1=LIFO) or against the oldest-to-newest account holdings (0=FIFO). |

**Table: TRADE\_HISTORY**

**Description:** The table contains the history of each trade transaction through the various states.

**Table 28 TPC-E TRADE\_HISTORY table description**

| Field Name | Data Type    | Constraints | Description                                                                                                                                                     |
|------------|--------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TH_T_ID    | BIGINT       | Not Null    | Trade identifier. This value will be used for the corresponding T_ID in the TRADE and SE_T_ID in the SETTLEMENT table if this trade request results in a trade. |
| TH_DTS     | DATETIME2(0) | Not Null    | Timestamp of when the trade history was updated.                                                                                                                |
| TH_ST_ID   | VARCHAR(4)   | Not Null    | Status type identifier.                                                                                                                                         |

**Table: TRADE\_REQUEST**

**Description:** The table contains information about pending limit trades that are waiting for a certain security price before the trades are submitted to the market.

**Table 29 TPC-E TRADE\_REQUEST table description**

| Field Name   | Data Type     | Constraints     | Description                                                                                                                                       |
|--------------|---------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| TR_T_ID      | BIGINT        | Not Null        | Trade request identifier. This value will be used for processing the pending limit order when it is subsequently triggered.                       |
| TR_TT_ID     | VARCHAR(3)    | Not Null        | Trade request type identifier. Identifies the type of trade.                                                                                      |
| TR_S_SYMB    | VARCHAR(15)   | Not Null        | Security symbol of the security the customer wants to trade.                                                                                      |
| TR_QTY       | INT           | Not Null<br>> 0 | Quantity of security the customer had requested to trade.                                                                                         |
| TR_BID_PRICE | DECIMAL(8, 2) | Not Null<br>> 0 | Price the customer wants per unit of security that they want to trade. Value of zero implies the customer wants to trade now at the market price. |
| TR_B_ID      | BIGINT        | Not Null        | Identifies the broker handling the trade.                                                                                                         |



**Table: TRADE\_TYPE**

**Description:** The table contains a list of valid trade types. There are two main categories of trades, buy and sell.

**Table 30 TPC-E TRADE\_TYPE table description**

| Field Name | Data Type   | Constraints        | Description                                                                                                                        |
|------------|-------------|--------------------|------------------------------------------------------------------------------------------------------------------------------------|
| TT_ID      | VARCHAR(3)  | Not Null           | Trade type identifier: Values are: "TMB", "TMS", "TSL", "TLS", and "TLB".                                                          |
| TT_NAME    | VARCHAR(12) | Not Null           | Trade type name. Examples "Limit Buy", "Limit Sell", "Market Buy", "Market Sell", "Stop Loss".                                     |
| TT_IS_SELL | BIT         | Not Null<br>in 0,1 | 1 if this is a "Sell" type transaction. 0 if this is a "Buy" type transaction.                                                     |
| TT_IS_MRKT | BIT         | Not Null<br>in 0,1 | 1 if this is a market transaction that is submitted to the market exchange emulator immediately. 0 if this is a limit transaction. |

**Table: WATCH\_ITEM**

**Description:** Each customer can create a list of securities that should be monitored. For every security that should be watched, one record is inserted into this table.

**Table 31 TPC-E WATCH\_ITEM table description**

| Field Name | Data Type   | Constraints | Description                      |
|------------|-------------|-------------|----------------------------------|
| WI_WL_ID   | BIGINT      | Not Null    | Watch list identifier.           |
| WI_S_SYMB  | VARCHAR(15) | Not Null    | Symbol of the security to watch. |

**Table: WATCH\_LIST**

**Description:** Each customer can create a list of securities that should be monitored. A customer can have more than one list. For each list, one record is inserted into the table.

**Table 32 TPC-E WATCH\_LIST table description**

| Field Name | Data Type | Constraints | Description                                         |
|------------|-----------|-------------|-----------------------------------------------------|
| WL_ID      | BIGINT    | Not Null    | Watch list identifier.                              |
| WL_C_ID    | BIGINT    | Not Null    | Identifier of customer who created this watch list. |

**Table:** ZIP\_CODE

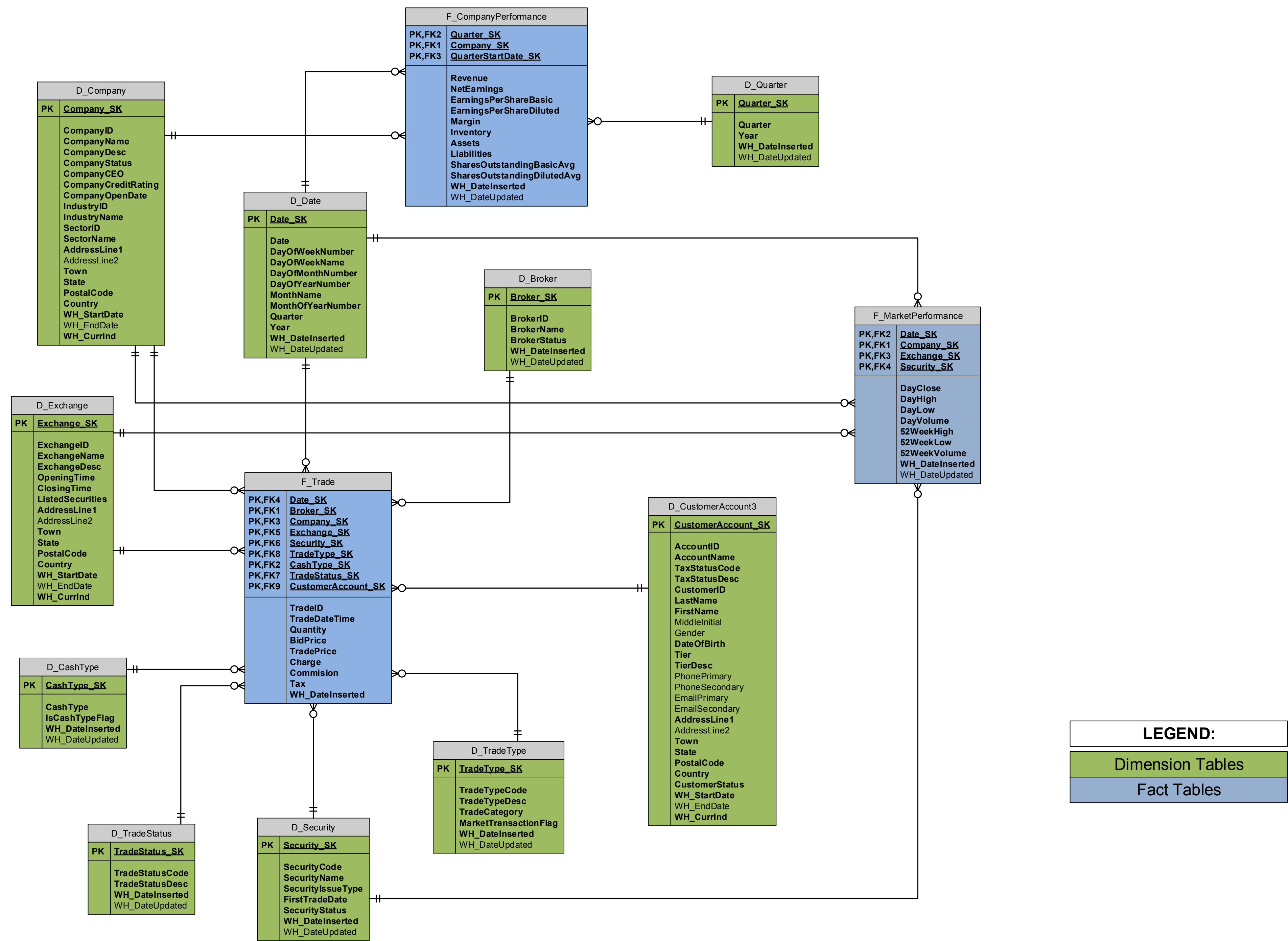
**Description:** The table contains zip and postal codes, towns, and divisions that go with them.

***Table 33 TPC-E ZIP\_CODE table description***

| Field Name | Data Type   | Constraints | Description                  |
|------------|-------------|-------------|------------------------------|
| ZC_CODE    | VARCHAR(12) | Not Null    | Postal code.                 |
| ZC_TOWN    | VARCHAR(80) | Not Null    | Town.                        |
| ZC_DIV     | VARCHAR(80) | Not Null    | State or province or county. |

Appendix E - Full Sized TPC-E Dimensional Model

The insert below is a larger version of the TPC-E dimensional model ERD as shown in chapter 4.



# Appendix F - Van Zyl Kruger's Dimensional Model Review

Below is a copy of the data vault model review report from Van Zyl Kruger, a recognised expert in the field of business Intelligence, dimensional modelling and statistics. Kruger's review was based on the following three artefacts:

- TPC-E ERD (Appendix C - Full Sized TPC-E ERD)
- Source-to-target mapping (Appendix G - Dimensional Source-to-Target Mapping)
- Dimensional model ERD (Appendix E - Full Sized TPC-E Dimensional Model)

## A. REVIEWER'S CREDENTIALS

### 1. DETAILS

1.1. Name: Van Zyl Kruger

### 2. RELEVANT ACADEMIC QUALIFICATIONS

- 2.1. B.Com (Actuarial Science) - University of Pretoria 1995 (ZAR)
- 2.2. B.Com (Hons) Mathematical Statistics – University of Pretoria 1996 (ZAR)
- 2.3. Graduate Statistician – Royal Statistical Society 1998 (UK)
- 2.4. M.Sc. (Business Intelligence) – Sheffield Hallam University 2010 (UK)

### 3. PROFESSIONAL MEMBERSHIPS

3.1. Fellow of the Royal Statistical Society

## B. RELEVANT EXPERIENCE

The reviewer has been a Business Intelligence and Data Warehousing professional since 1996 and has been personally involved in the delivery of over 20 data warehousing data models throughout different industries across the globe in his career. He is currently employed at Altis Consulting in Sydney Australia as a Senior Business Intelligence Consultant.

Some of the major Data Warehousing modelling projects include:

- Medical Services Organisation Health Insurance Data Warehouse (ZAR)
- Fleming Asset Management Trading and Rating Data Warehouse (ZAR)
- Vodacom Billing Data Warehouse (ZAR)
- Vodacom Out-Of-Box Failures Data Warehouse (ZAR)
- PrimeCure Data Warehouse (ZAR)

- MIP Holdings Health Insurance Data Warehouse (ZAR)
- Netcare United Kingdom Health Outcomes Data Warehouse (UK)
- Cato Research Data Warehouse (USA, North Carolina)
- MTN Group (Dubai)
- BHP Billiton Manganese (Australia and ZAR)
- Catholic Healthcare (Australia)
- DP World (Australia)

## **C. FINDINGS**

### **1. GENERAL**

The model theoretically follows a typical “star schema”, Ralph Kimball or Multi-dimensional design. The model contains all the typical components expected in this type of design, Facts and Dimensions (Type I and Type II).

The model looks theoretically sound and without any major defects or shortcomings (attributes and possible improvements discussed in Sections 2 and 3). More important, the model will provide a good data model (or source) for most of the major reporting and Business Intelligence toolsets.

### **2. TABLES**

- Fact tables (F\_). The fact tables are not all aggregate-able and it might be a good idea to include an abbreviation in the table name to describe the type of the fact.
- Dimension tables (D\_). Both Type I and Type II tables are used in the design and adding an abbreviation in the table name to explain the Type used will make usage of the model simpler.
- I question the usage of using both a Last Indicator and Start –and End Dates in Type II dimensions. This is not just personal preference, a number of Integration tools do not support the maintenance of both and it could potentially increase development effort to have both without much benefit.

### **3. COSMETICS**

- Use underscores throughout, your keys contain them XXXX\_SK but your metrics don't.

# Appendix G - Dimensional Source-to-Target Mapping

This appendix contains the source-to-target mapping that was used to implement the dimensional model. Table 1 to

table 8 below shows the source-to-target mappings for all the dimension tables.

Table 9 to



table 11 below shows the source-to-target mappings for the three fact tables.

**Table 1 Broker dimension (D\_Broker) source-to-target mapping**

| Target           |              |             |        | Source        |             |         |              |             | Transformation                                                 |
|------------------|--------------|-------------|--------|---------------|-------------|---------|--------------|-------------|----------------------------------------------------------------|
| Field            | Data Type    | Allow Nulls | SCD    | Source System | Table       | Field   | Data Type    | Allow Nulls |                                                                |
| Broker_SK        | INT          | N           |        | Derived       |             |         | INT          | N           | Surrogate key generated by DBMS. IDENTITY(1,1)                 |
| BrokerID         | BIGINT       | N           | Lookup | TPCE          | BROKER      | B_ID    | BIGINT       | N           |                                                                |
| BrokerName       | VARCHAR(49)  | N           | 1      | TPCE          | BROKER      | B_NAME  | VARCHAR(49)  | N           |                                                                |
| BrokerStatusCode | VARCHAR(4)   | N           | 1      | TPCE          | STATUS_TYPE | ST_ID   | VARCHAR(4)   | N           |                                                                |
| BrokerStatusDesc | VARCHAR(10)  | N           | 1      | TPCE          | STATUS_TYPE | ST_NAME | VARCHAR(10)  | N           |                                                                |
| WH_DateInserted  | DATETIME2(0) | N           |        | Derived       |             |         | DATETIME2(0) | N           | GETDATE when record is inserted                                |
| WH_DateUpdated   | DATETIME2(0) | Y           |        | Derived       |             |         | DATETIME2(0) | Y           | NULL when record is inserted<br>GETDATE when record is updated |

**Table 2 Cash type dimension (D\_CashType) source-to-target mapping**

| Target          |              |             |        | Source        |       |           |              |             | Transformation                                                 |
|-----------------|--------------|-------------|--------|---------------|-------|-----------|--------------|-------------|----------------------------------------------------------------|
| Field           | Data Type    | Allow Nulls | SCD    | Source System | Table | Field     | Data Type    | Allow Nulls |                                                                |
| CashType_SK     | INT          | N           |        | Derived       |       |           | INT          | N           | Surrogate key generated by DBMS. IDENTITY(1,1)                 |
| CashType        | VARCHAR(12)  | N           | 1      | TPCE          | TRADE | T_IS_CASH | bit          | N           | IF T_IS_CASH = 1 THEN<br>'Cash Account'<br>ELSE<br>'Margin'    |
| IsCashFlag      | bit          | N           | Lookup | TPCE          | TRADE | T_IS_CASH | bit          | N           |                                                                |
| WH_DateInserted | DATETIME2(0) | N           |        | Derived       |       |           | DATETIME2(0) | N           | GETDATE when record is inserted                                |
| WH_DateUpdated  | DATETIME2(0) | Y           |        | Derived       |       |           | DATETIME2(0) | Y           | NULL when record is inserted<br>GETDATE when record is updated |

**Table 3 Company dimension (D\_Company) source-to-target mapping**

| Target              |              |             |        | Source        |             |              |              |             | Transformation                                 |
|---------------------|--------------|-------------|--------|---------------|-------------|--------------|--------------|-------------|------------------------------------------------|
| Field               | Data Type    | Allow Nulls | SCD    | Source System | Table       | Field        | Data Type    | Allow Nulls |                                                |
| Company_SK          | INT          | N           |        | Derived       |             |              | INT          | N           | Surrogate key generated by DBMS. IDENTITY(1,1) |
| CompanyID           | BIGINT       | N           | Lookup | TPCE          | COMPANY     | CO_ID        | BIGINT       | N           |                                                |
| CompanyName         | VARCHAR(60)  | N           | 1      | TPCE          | COMPANY     | CO_NAME      | VARCHAR(60)  | N           |                                                |
| CompanyDesc         | VARCHAR(150) | N           | 1      | TPCE          | COMPANY     | CO_DESC      | VARCHAR(150) | N           |                                                |
| CompanyStatusCode   | VARCHAR(4)   | N           | 1      | TPCE          | STATUS_TYPE | ST_ID        | VARCHAR(4)   | N           |                                                |
| CompanyStatusDesc   | VARCHAR(10)  | N           | 1      | TPCE          | STATUS_TYPE | ST_NAME      | VARCHAR(10)  | N           |                                                |
| CompanyCEO          | VARCHAR(46)  | N           | 2      | TPCE          | COMPANY     | CO_CEO       | VARCHAR(46)  | N           |                                                |
| CompanyCreditRating | VARCHAR(4)   | N           | 2      | TPCE          | COMPANY     | CO_SP_RATE   | VARCHAR(4)   | N           |                                                |
| CompanyOpenDate     | date         | N           | 1      | TPCE          | COMPANY     | CO_OPEN_DATE | date         | N           |                                                |
| IndustryCode        | VARCHAR(2)   | N           | 1      | TPCE          | COMPANY     | CO_IN_ID     | VARCHAR(2)   | N           |                                                |
| IndustryDesc        | VARCHAR(50)  | N           | 1      | TPCE          | INDUSTRY    | IN_NAME      | VARCHAR(50)  | N           |                                                |
| SectorCode          | VARCHAR(2)   | N           | 1      | TPCE          | INDUSTRY    | IN_SC_ID     | VARCHAR(2)   | N           |                                                |

| Target            |              |             |     | Source        |          |            |              |             | Transformation                                                                                                           |
|-------------------|--------------|-------------|-----|---------------|----------|------------|--------------|-------------|--------------------------------------------------------------------------------------------------------------------------|
| Field             | Data Type    | Allow Nulls | SCD | Source System | Table    | Field      | Data Type    | Allow Nulls |                                                                                                                          |
| SectorDesc        | VARCHAR(30)  | N           | 1   | TPCE          | SECTOR   | SC_ID      | VARCHAR(2)   | N           |                                                                                                                          |
| AddressLine1      | VARCHAR(80)  | N           | 1   | TPCE          | ADDRESS  | AD_LINE1   | VARCHAR(80)  | N           |                                                                                                                          |
| AddressLine2      | VARCHAR(80)  | N           | 1   | TPCE          | ADDRESS  | AD_LINE2   | VARCHAR(80)  | Y           | IF ADDRESS.AD_LINE2 IS NULL THEN<br>"<br>ELSE<br>ADDRESS.AD_LINE2                                                        |
| TownName          | VARCHAR(80)  | N           | 1   | TPCE          | ZIP_CODE | ZC_TOWN    | VARCHAR(80)  | N           |                                                                                                                          |
| StateProvinceCode | VARCHAR(80)  | N           | 1   | TPCE          | ZIP_CODE | ZC_DIV     | VARCHAR(80)  | N           |                                                                                                                          |
| PostalCode        | VARCHAR(12)  | N           | 1   | TPCE          | ADDRESS  | AD_ZC_CODE | VARCHAR(12)  | N           |                                                                                                                          |
| CountryName       | VARCHAR(80)  | N           | 1   | TPCE          | ADDRESS  | AD_CTRY    | VARCHAR(80)  | Y           | Only USA in uppercase. The rest in init caps.<br>IF ADDRESS.AD_CTRY IS NULL THEN<br>'Unknown'<br>ELSE<br>ADDRESS.AD_CTRY |
| WH_StartDate      | DATETIME2(0) | N           |     | Derived       |          |            | DATETIME2(0) | N           | GETDATE when record is inserted.                                                                                         |
| WH_EndDate        | DATETIME2(0) | Y           |     | Derived       |          |            | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated because of Type 2 SCD.                               |
| WH_CurrInd        | bit          | N           |     | Derived       |          |            | bit          | N           | 1 if record is not end-dated.<br>0 if record is end-dated.                                                               |

**Table 4 Customer account dimension (D\_CustomerAccount) source-to-target mapping**

| Target             |             |             |        | Source        |                  |           |             |             | Transformation                                                                                                                                                                                                                           |
|--------------------|-------------|-------------|--------|---------------|------------------|-----------|-------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Field              | Data Type   | Allow Nulls | SCD    | Source System | Table            | Field     | Data Type   | Allow Nulls |                                                                                                                                                                                                                                          |
| CustomerAccount_SK | INT         | N           |        | Derived       |                  |           | INT         | N           | Surrogate key generated by DBMS. IDENTITY(1,1)                                                                                                                                                                                           |
| AccountID          | BIGINT      | N           | Lookup | TPCE          | CUSTOMER_ACCOUNT | CA_ID     | BIGINT      | N           |                                                                                                                                                                                                                                          |
| AccountName        | VARCHAR(50) | N           | 1      | TPCE          | CUSTOMER_ACCOUNT | CA_NAME   | VARCHAR(50) | N           |                                                                                                                                                                                                                                          |
| TaxStatusCode      | INT         | N           | 1      | TPCE          | CUSTOMER_ACCOUNT | CA_TAX_ST | TINYINT     | N           |                                                                                                                                                                                                                                          |
| TaxStatusDesc      | VARCHAR(30) | N           | 1      | TPCE          | CUSTOMER_ACCOUNT | CA_TAX_ST | TINYINT     | N           | IF CUSTOMER_ACCOUNT.CA_TAX_ST = 0 THEN<br>'Not taxable.'<br>ELSEIF CUSTOMER_ACCOUNT.CA_TAX_ST = 1 THEN<br>'Taxable. Withhold tax.'<br>ELSEIF CUSTOMER_ACCOUNT.CA_TAX_ST = 2 THEN<br>'Taxable. Do not withhold tax.'<br>ELSE<br>'Unknown' |
| CustomerID         | BIGINT      | N           | 1      | TPCE          | CUSTOMER         | C_ID      | BIGINT      | N           |                                                                                                                                                                                                                                          |
| LastName           | VARCHAR(25) | N           | 1      | TPCE          | CUSTOMER         | C_L_NAME  | VARCHAR(25) | N           |                                                                                                                                                                                                                                          |
| FirstName          | VARCHAR(20) | N           | 1      | TPCE          | CUSTOMER         | C_F_NAME  | VARCHAR(20) | N           |                                                                                                                                                                                                                                          |
| MiddleInitial      | char(1)     | N           | 1      | TPCE          | CUSTOMER         | C_M_NAME  | char(1)     | Y           | IF CUSTOMER.C_M_NAME IS NULL THEN<br>"<br>ELSE<br>CUSTOMER.C_M_NAME                                                                                                                                                                      |
| Gender             | VARCHAR(7)  | N           | 1      | TPCE          | CUSTOMER         | C_GNDR    | VARCHAR(1)  | Y           | IF CUSTOMER.C_GNDR = 'M' THEN<br>'Male'<br>ELSEIF CUSTOMER.C_GNDR = 'F' THEN<br>'Female'<br>ELSE<br>'Unknown'                                                                                                                            |
| DateOfBirth        | date        | N           | 1      | TPCE          | CUSTOMER         | C_DOB     | date        | N           |                                                                                                                                                                                                                                          |
| TierNumber         | INT         | N           | 2      | TPCE          | CUSTOMER         | C_TIER    | TINYINT     | N           |                                                                                                                                                                                                                                          |
| TierDesc           | VARCHAR(35) | N           | 2      | TPCE          | CUSTOMER         | C_TIER    | TINYINT     | N           | IF CUSTOMER.C_TIER = 1 THEN<br>'Has between 1 and 4 accounts.'<br>ELSEIF CUSTOMER.C_TIER = 2 THEN<br>'Has between 2 and 8 accounts.'<br>ELSEIF CUSTOMER.C_TIER = 3 THEN<br>'Has between 5 and 10 accounts.'<br>ELSE<br>'Unknown'         |

| Target            |             |             |     | Source        |          |            |             |             | Transformation                                                                                                                                                                                                                                                                              |
|-------------------|-------------|-------------|-----|---------------|----------|------------|-------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Field             | Data Type   | Allow Nulls | SCD | Source System | Table    | Field      | Data Type   | Allow Nulls |                                                                                                                                                                                                                                                                                             |
| PhonePrimary      | VARCHAR(30) | N           | 1   | TPCE          | CUSTOMER | C_CTRY_1   | VARCHAR(3)  | Y           | IF C_CTRY_1 AND C_AREA_1 AND C_LOCAL_1 IS NOT NULL THEN<br>IF C_EXT_1 IS NULL THEN<br>CONCAT(C_CTRY_1,'-',C_AREA_1,'-',LEFT(C_LOCAL_1,3),'-',RIGHT(C_LOCAL_1,4))<br>ELSE<br>CONCAT(C_CTRY_1,'-',C_AREA_1,'-',LEFT(C_LOCAL_1,3),'-',RIGHT(C_LOCAL_1,4),' ext.',C_EXT_1)<br>ELSE<br>'Unknown' |
|                   |             |             |     | TPCE          | CUSTOMER | C_AREA_1   | VARCHAR(3)  | Y           |                                                                                                                                                                                                                                                                                             |
|                   |             |             |     | TPCE          | CUSTOMER | C_LOCAL_1  | VARCHAR(10) | Y           |                                                                                                                                                                                                                                                                                             |
|                   |             |             |     | TPCE          | CUSTOMER | C_EXT_1    | VARCHAR(5)  | Y           |                                                                                                                                                                                                                                                                                             |
| PhoneSecondary    | VARCHAR(30) | N           | 1   | TPCE          | CUSTOMER | C_CTRY_2   | VARCHAR(3)  | Y           | IF C_CTRY_2 AND C_AREA_2 AND C_LOCAL_2 IS NOT NULL THEN<br>IF C_EXT_2 IS NULL THEN<br>CONCAT(C_CTRY_2,'-',C_AREA_2,'-',LEFT(C_LOCAL_2,3),'-',RIGHT(C_LOCAL_2,4))<br>ELSE<br>CONCAT(C_CTRY_2,'-',C_AREA_2,'-',LEFT(C_LOCAL_2,3),'-',RIGHT(C_LOCAL_2,4),' ext.',C_EXT_2)<br>ELSE<br>'Unknown' |
|                   |             |             |     | TPCE          | CUSTOMER | C_AREA_2   | VARCHAR(3)  | Y           |                                                                                                                                                                                                                                                                                             |
|                   |             |             |     | TPCE          | CUSTOMER | C_LOCAL_2  | VARCHAR(10) | Y           |                                                                                                                                                                                                                                                                                             |
|                   |             |             |     | TPCE          | CUSTOMER | C_EXT_2    | VARCHAR(5)  | Y           |                                                                                                                                                                                                                                                                                             |
| EmailPrimary      | VARCHAR(50) | N           | 1   | TPCE          | CUSTOMER | C_EMAIL_1  | VARCHAR(50) | Y           | IF C_EMAIL1 IS NULL THEN<br>'Unknown'<br>ELSE<br>C_EMAIL1                                                                                                                                                                                                                                   |
| EmailSecondary    | VARCHAR(50) | N           | 1   | TPCE          | CUSTOMER | C_EMAIL_2  | VARCHAR(50) | Y           | IF C_EMAIL2 IS NULL THEN<br>'Unknown'<br>ELSE<br>C_EMAIL2                                                                                                                                                                                                                                   |
| AddressLine1      | VARCHAR(80) | N           | 1   | TPCE          | ADDRESS  | AD_LINE1   | VARCHAR(80) | N           |                                                                                                                                                                                                                                                                                             |
| AddressLine2      | VARCHAR(80) | Y           | 1   | TPCE          | ADDRESS  | AD_LINE2   | VARCHAR(80) | Y           | IF ADDRESS.AD_LINE2 IS NULL THEN<br>"<br>ELSE<br>ADDRESS.AD_LINE2                                                                                                                                                                                                                           |
| TownName          | VARCHAR(80) | N           | 1   | TPCE          | ZIP_CODE | ZC_TOWN    | VARCHAR(80) | N           |                                                                                                                                                                                                                                                                                             |
| StateProvinceCode | VARCHAR(80) | N           | 1   | TPCE          | ZIP_CODE | ZC_DIV     | VARCHAR(80) | N           |                                                                                                                                                                                                                                                                                             |
| PostalCode        | VARCHAR(12) | N           | 1   | TPCE          | ADDRESS  | AD_ZC_CODE | VARCHAR(12) | N           |                                                                                                                                                                                                                                                                                             |

| Target             |              |             |     | Source        |             |         |              |             | Transformation                                                                                                           |
|--------------------|--------------|-------------|-----|---------------|-------------|---------|--------------|-------------|--------------------------------------------------------------------------------------------------------------------------|
| Field              | Data Type    | Allow Nulls | SCD | Source System | Table       | Field   | Data Type    | Allow Nulls |                                                                                                                          |
| CountryName        | VARCHAR(80)  | N           | 1   | TPCE          | ADDRESS     | AD_CTRY | VARCHAR(80)  | Y           | Only USA in uppercase. The rest in init caps.<br>IF ADDRESS.AD_CTRY IS NULL THEN<br>'Unknown'<br>ELSE<br>ADDRESS.AD_CTRY |
| CustomerStatusCode | VARCHAR(4)   | N           | 1   | TPCE          | STATUS_TYPE | ST_ID   | VARCHAR(4)   | N           |                                                                                                                          |
| CustomerStatusDesc | VARCHAR(10)  | N           | 1   | TPCE          | STATUS_TYPE | ST_NAME | VARCHAR(10)  | N           |                                                                                                                          |
| WH_StartDate       | DATETIME2(7) | N           |     | Derived       |             |         | DATETIME2(0) | N           | GETDATE when record is inserted.                                                                                         |
| WH_EndDate         | DATETIME2(7) | Y           |     | Derived       |             |         | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated because of Type 2 SCD.                               |
| WH_CurrInd         | bit          | N           |     | Derived       |             |         | bit          | N           | 1 if record is not end-dated.<br>0 if record is end-dated.                                                               |

**Table 5 Exchange dimension (D\_Exchange) source-to-target mapping**

| Target            |              |             |        | Source        |          |             |              |             | Transformation                                                                                                           |
|-------------------|--------------|-------------|--------|---------------|----------|-------------|--------------|-------------|--------------------------------------------------------------------------------------------------------------------------|
| Field             | Data Type    | Allow Nulls | SCD    | Source System | Table    | Field       | Data Type    | Allow Nulls |                                                                                                                          |
| Exchange_SK       | INT          | N           |        | Derived       |          |             | INT          | N           | Surrogate key generated by DBMS. IDENTITY(1,1)                                                                           |
| ExchangeID        | VARCHAR(6)   | N           | Lookup | TPCE          | EXCHANGE | EX_ID       | VARCHAR(6)   | N           |                                                                                                                          |
| ExchangeName      | VARCHAR(100) | N           | 1      | TPCE          | EXCHANGE | EX_NAME     | VARCHAR(100) | N           |                                                                                                                          |
| ExchangeDesc      | VARCHAR(150) | N           | 1      | TPCE          | EXCHANGE | EX_DESC     | VARCHAR(150) | N           |                                                                                                                          |
| OpeningTime       | time(0)      | N           | 2      | TPCE          | EXCHANGE | EX_OPEN     | SMALLINT     | N           |                                                                                                                          |
| ClosingTime       | time(0)      | N           | 2      | TPCE          | EXCHANGE | EX_CLOSE    | SMALLINT     | N           |                                                                                                                          |
| ListedSecurities  | INT          | N           | 1      | TPCE          | EXCHANGE | EX_NUM_SYMB | INT          | N           |                                                                                                                          |
| AddressLine1      | VARCHAR(80)  | N           | 1      | TPCE          | ADDRESS  | AD_LINE1    | VARCHAR(80)  | N           |                                                                                                                          |
| AddressLine2      | VARCHAR(80)  | N           | 1      | TPCE          | ADDRESS  | AD_LINE2    | VARCHAR(80)  | Y           | IF ADDRESS.AD_LINE2 IS NULL THEN<br>"<br>ELSE<br>ADDRESS.AD_LINE2                                                        |
| TownName          | VARCHAR(80)  | N           | 1      | TPCE          | ZIP_CODE | ZC_TOWN     | VARCHAR(80)  | N           |                                                                                                                          |
| StateProvinceCode | VARCHAR(80)  | N           | 1      | TPCE          | ZIP_CODE | ZC_DIV      | VARCHAR(80)  | N           |                                                                                                                          |
| PostalCode        | VARCHAR(12)  | N           | 1      | TPCE          | ADDRESS  | AD_ZC_CODE  | VARCHAR(12)  | N           |                                                                                                                          |
| CountryName       | VARCHAR(80)  | N           | 1      | TPCE          | ADDRESS  | AD_CTRY     | VARCHAR(80)  | Y           | Only USA in uppercase. The rest in init caps.<br>IF ADDRESS.AD_CTRY IS NULL THEN<br>'Unknown'<br>ELSE<br>ADDRESS.AD_CTRY |
| WH_StartDate      | DATETIME2(0) | N           |        | Derived       |          |             | DATETIME2(0) | N           | GETDATE when record is inserted.                                                                                         |
| WH_EndDate        | DATETIME2(0) | Y           |        | Derived       |          |             | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated because of Type 2 SCD.                               |
| WH_CurrInd        | bit          | N           |        | Derived       |          |             | bit          | N           | 1 if record is not end-dated.<br>0 if record is end-dated.                                                               |

**Table 6 Security dimension (D\_Security) source-to-target mapping**

| Target             |              |             |        | Source        |             |              |              |             | Transformation                                                 |
|--------------------|--------------|-------------|--------|---------------|-------------|--------------|--------------|-------------|----------------------------------------------------------------|
| Field              | Data Type    | Allow Nulls | SCD    | Source System | Table       | Field        | Data Type    | Allow Nulls |                                                                |
| Security_SK        | INT          | N           |        | Derived       |             |              | INT          | N           | Surrogate key generated by DBMS. IDENTITY(1,1)                 |
| SecurityCode       | VARCHAR(15)  | N           | Lookup | TPCE          | SECURITY    | S_SYMB       | VARCHAR(15)  | N           |                                                                |
| SecurityName       | VARCHAR(70)  | N           | 1      | TPCE          | SECURITY    | S_NAME       | VARCHAR(70)  | N           |                                                                |
| SecurityIssueType  | VARCHAR(6)   | N           | 1      | TPCE          | SECURITY    | S_ISSUE      | VARCHAR(6)   | N           |                                                                |
| FirstTradeDate     | date         | N           | 1      | TPCE          | SECURITY    | S_START_DATE | date         | N           |                                                                |
| SecurityStatusCode | VARCHAR(4)   | N           | 1      | TPCE          | STATUS_TYPE | ST_ID        | VARCHAR(4)   | N           |                                                                |
| SecurityStatusDesc | VARCHAR(10)  | N           | 1      | TPCE          | STATUS_TYPE | ST_NAME      | VARCHAR(10)  | N           |                                                                |
| WH_DateInserted    | DATETIME2(0) | N           |        | Derived       |             |              | DATETIME2(0) | N           | GETDATE when record is inserted                                |
| WH_DateUpdated     | DATETIME2(0) | Y           |        | Derived       |             |              | DATETIME2(0) | Y           | NULL when record is inserted<br>GETDATE when record is updated |

**Table 7 Trade status dimension (D\_TradeStatus) source-to-target mapping**

| Target          |              |             |        | Source        |             |         |              |             | Transformation                                                 |
|-----------------|--------------|-------------|--------|---------------|-------------|---------|--------------|-------------|----------------------------------------------------------------|
| Field           | Data Type    | Allow Nulls | SCD    | Source System | Table       | Field   | Data Type    | Allow Nulls |                                                                |
| TradeStatus_SK  | INT          | N           |        | Derived       |             |         | INT          | N           | Surrogate key generated by DBMS. IDENTITY(1,1)                 |
| TradeStatusCode | VARCHAR(4)   | N           | Lookup | TPCE          | STATUS_TYPE | ST_ID   | VARCHAR(4)   | N           |                                                                |
| TradeStatusDesc | VARCHAR(10)  | N           | 1      | TPCE          | STATUS_TYPE | ST_NAME | VARCHAR(10)  | N           |                                                                |
| WH_DateInserted | DATETIME2(0) | N           |        | Derived       |             |         | DATETIME2(0) | N           | GETDATE when record is inserted                                |
| WH_DateUpdated  | DATETIME2(0) | Y           |        | Derived       |             |         | DATETIME2(0) | Y           | NULL when record is inserted<br>GETDATE when record is updated |



**Table 8 Trade type dimension (D\_TradeType) source-to-target mapping**

| Target                |              |             |        | Source        |            |            |              |             | Transformation                                                 |
|-----------------------|--------------|-------------|--------|---------------|------------|------------|--------------|-------------|----------------------------------------------------------------|
| Field                 | Data Type    | Allow Nulls | SCD    | Source System | Table      | Field      | Data Type    | Allow Nulls |                                                                |
| TradeType_SK          | INT          | N           |        | Derived       |            |            | INT          | N           | Surrogate key generated by DBMS. IDENTITY(1,1)                 |
| TradeTypeCode         | VARCHAR(3)   | N           | Lookup | TPCE          | TRADE_TYPE | TT_ID      | VARCHAR(3)   | N           |                                                                |
| TradeTypeDesc         | VARCHAR(12)  | N           | 1      | TPCE          | TRADE_TYPE | TT_NAME    | VARCHAR(12)  | N           |                                                                |
| TradeCategory         | VARCHAR(4)   | N           | 1      | TPCE          | TRADE_TYPE | TT_IS_SELL | bit          | N           | IF TRADE_TYPE.TT_IS_SELL = 0 THEN<br>'Buy'<br>ELSE<br>'Sell'   |
| MarketTransactionFlag | bit          | N           | 1      | TPCE          | TRADE_TYPE | TT_IS_MRKT | bit          | N           |                                                                |
| WH_DateInserted       | DATETIME2(0) | N           |        | Derived       |            |            | DATETIME2(0) | N           | GETDATE when record is inserted                                |
| WH_DateUpdated        | DATETIME2(0) | Y           |        | Derived       |            |            | DATETIME2(0) | Y           | NULL when record is inserted<br>GETDATE when record is updated |

**Table 9 Quarterly company performance fact (F\_QuarterlyCompanyPerformance) source-to-target mapping**

| Target                      |              |             | Source        |           |                   |               |             | Transformation                                                  |
|-----------------------------|--------------|-------------|---------------|-----------|-------------------|---------------|-------------|-----------------------------------------------------------------|
| Field                       | Data Type    | Allow Nulls | Source System | Table     | Field             | Data Type     | Allow Nulls |                                                                 |
| FiscalQuarter_SK            | INT          | N           | TPCE          | FINANCIAL | FI_YEAR           | SMALLINT      | N           | (FI_YEAR * 10) + FI_QTR                                         |
|                             |              |             | TPCE          | FINANCIAL | FI_QTR            | TINYINT       | N           |                                                                 |
| Company_SK                  | INT          | N           | TPCE_DM       | D_Company | Company_SK        | INT           | N           |                                                                 |
| QuarterStartDate_SK         | INT          | N           | TPCE          | FINANCIAL | FI_QTR_START_DATE | date          | N           | CONVERT Date to 8 digit INTegeR.<br>E.g. 2011/08/20 to 20110820 |
| Revenue                     | money        | N           | TPCE          | FINANCIAL | FI_REVENUE        | DECIMAL(15,2) | N           |                                                                 |
| NetEarnings                 | money        | N           | TPCE          | FINANCIAL | FI_NET_EARN       | DECIMAL(15,2) | N           |                                                                 |
| EarningsPerShareBasic       | money        | N           | TPCE          | FINANCIAL | FI_BASIC_EPS      | DECIMAL(10,2) | N           |                                                                 |
| EarningsPerShareDiluted     | money        | N           | TPCE          | FINANCIAL | FI_DILUT_EPS      | DECIMAL(10,2) | N           |                                                                 |
| Margin                      | money        | N           | TPCE          | FINANCIAL | FI_MARGIN         | DECIMAL(10,2) | N           |                                                                 |
| Inventory                   | money        | N           | TPCE          | FINANCIAL | FI_INVENTORY      | DECIMAL(15,2) | N           |                                                                 |
| Assets                      | money        | N           | TPCE          | FINANCIAL | FI_ASSETS         | DECIMAL(15,2) | N           |                                                                 |
| Liabilities                 | money        | N           | TPCE          | FINANCIAL | FI_LIABILITY      | DECIMAL(15,2) | N           |                                                                 |
| SharesOutstandingBasicAvg   | BIGINT       | N           | TPCE          | FINANCIAL | FI_OUT_BASIC      | BIGINT        | N           |                                                                 |
| SharesOutstandingDilutedAvg | BIGINT       | N           | TPCE          | FINANCIAL | FI_OUT_DILUT      | BIGINT        | N           |                                                                 |
| WH_DateInserted             | DATETIME2(0) | N           | Derived       |           |                   | DATETIME2(0)  | N           | GETDATE when record is inserted                                 |

**Table 10 Daily market performance fact (F\_DailyMarketPerformance) source-to-target mapping**

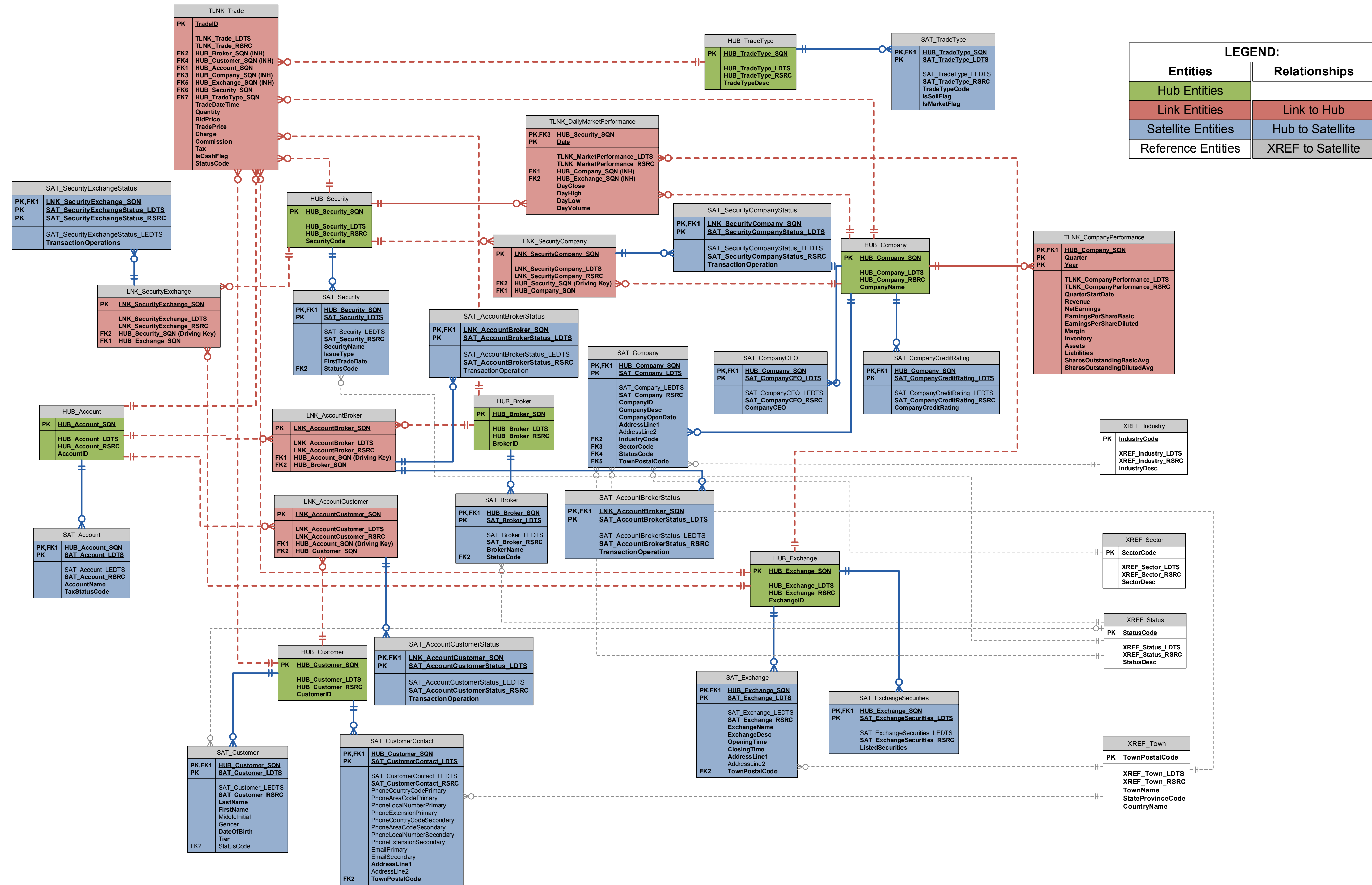
| Target          |              |             | Source        |              |             |              |             | Transformation                                                  |
|-----------------|--------------|-------------|---------------|--------------|-------------|--------------|-------------|-----------------------------------------------------------------|
| Field           | Data Type    | Allow Nulls | Source System | Table        | Field       | Data Type    | Allow Nulls |                                                                 |
| Date_SK         | INT          | N           | TPCE          | DAILY_MARKET | DM_DATE     | Date         | N           | CONVERT Date to 8 digit INTEger.<br>E.g. 2011/08/20 to 20110820 |
| Company_SK      | INT          | N           | TPCE_DM       | D_Company    | Company_SK  | INT          | N           |                                                                 |
| Exchange_SK     | INT          | N           | TPCE_DM       | D_Exchange   | Exchange_SK | INT          | N           |                                                                 |
| Security_SK     | INT          | N           | TPCE_DM       | D_Security   | Security_SK | INT          | N           |                                                                 |
| DayClose        | money        | N           | TPCE          | DAILY_MARKET | DM_CLOSE    | DECIMAL(8,2) | N           |                                                                 |
| DayHigh         | money        | N           | TPCE          | DAILY_MARKET | DM_HIGH     | DECIMAL(8,2) | N           |                                                                 |
| DayLow          | money        | N           | TPCE          | DAILY_MARKET | DM_LOW      | DECIMAL(8,2) | N           |                                                                 |
| DayVolume       | BIGINT       | N           | TPCE          | DAILY_MARKET | DM_VOL      | BIGINT       | N           |                                                                 |
| [52WeekHigh]    | money        | N           | TPCE          | DAILY_MARKET | DM_HIGH     | DECIMAL(8,2) | N           | MAX(DM_HIGH) for past 52 weeks                                  |
| [52WeekLow]     | money        | N           | TPCE          | DAILY_MARKET | DM_LOW      | DECIMAL(8,2) | N           | MIN(DM_LOW) for past 52 weeks                                   |
| [52WeekVolume]  | BIGINT       | N           | TPCE          | DAILY_MARKET | DM_VOL      | BIGINT       | N           | SUM(DM_VOL) for past 52 weeks                                   |
| WH_DateInserted | DATETIME2(0) | N           | Derived       |              |             | DATETIME2(0) | N           | GETDATE when record is inserted                                 |

**Table 11 Trade fact (F\_Trade) source-to-target mapping**

| Target             |              |             | Source        |                   |                    |               |             | Transformation                                                  |
|--------------------|--------------|-------------|---------------|-------------------|--------------------|---------------|-------------|-----------------------------------------------------------------|
| Field              | Data Type    | Allow Nulls | Source System | Table             | Field              | Data Type     | Allow Nulls |                                                                 |
| Date_SK            | INT          | N           | TPCE          | TRADE             | T_DTS              | DATETIME2(0)  | N           | CONVERT Date to 8 digit INTEger.<br>E.g. 2011/08/20 to 20110820 |
| Broker_SK          | INT          | N           | TPCE_DM       | D_Broker          | Broker_SK          | INT           | N           |                                                                 |
| CustomerAccount_SK | INT          | N           | TPCE_DM       | D_CustomerAccount | CustomerAccount_SK | INT           | N           |                                                                 |
| Company_SK         | INT          | N           | TPCE_DM       | D_Company         | Company_SK         | INT           | N           |                                                                 |
| Exchange_SK        | INT          | N           | TPCE_DM       | D_Exchange        | Exchange_SK        | INT           | N           |                                                                 |
| Security_SK        | INT          | N           | TPCE_DM       | D_Security        | Security_SK        | INT           | N           |                                                                 |
| TradeType_SK       | INT          | N           | TPCE_DM       | D_TradeType       | TradeType_SK       | INT           | N           |                                                                 |
| CashType_SK        | INT          | N           | TPCE_DM       | D_CashType        | CashType_SK        | INT           | N           |                                                                 |
| TradeStatus_SK     | INT          | N           | TPCE_DM       | D_TradeStatus     | TradeStatus_SK     | INT           | N           |                                                                 |
| TradeID            | BIGINT       | N           | TPCE          | TRADE             | T_ID               | BIGINT        | N           |                                                                 |
| TradeDATETIME      | DATETIME2(0) | N           | TPCE          | TRADE             | T_DTS              | DATETIME2(0)  | N           |                                                                 |
| Quantity           | INT          | N           | TPCE          | TRADE             | T_QTY              | INT           | N           |                                                                 |
| BidPrice           | money        | N           | TPCE          | TRADE             | T_BID_PRICE        | DECIMAL(8,2)  | N           |                                                                 |
| TradePrice         | money        | N           | TPCE          | TRADE             | T_TRADE_PRICE      | DECIMAL(8,2)  | N           |                                                                 |
| Charge             | money        | N           | TPCE          | TRADE             | T_CHRG             | DECIMAL(10,2) | N           |                                                                 |
| Commission         | money        | N           | TPCE          | TRADE             | T_COMM             | DECIMAL(10,2) | N           |                                                                 |
| Tax                | money        | N           | TPCE          | TRADE             | T_TAX              | DECIMAL(10,2) | N           |                                                                 |
| WH_DateInserted    | DATETIME2(0) | N           | Derived       |                   |                    | DATETIME2(0)  | N           | GETDATE when record is inserted                                 |

# Appendix H - Full Sized TPC-E Data Vault Model

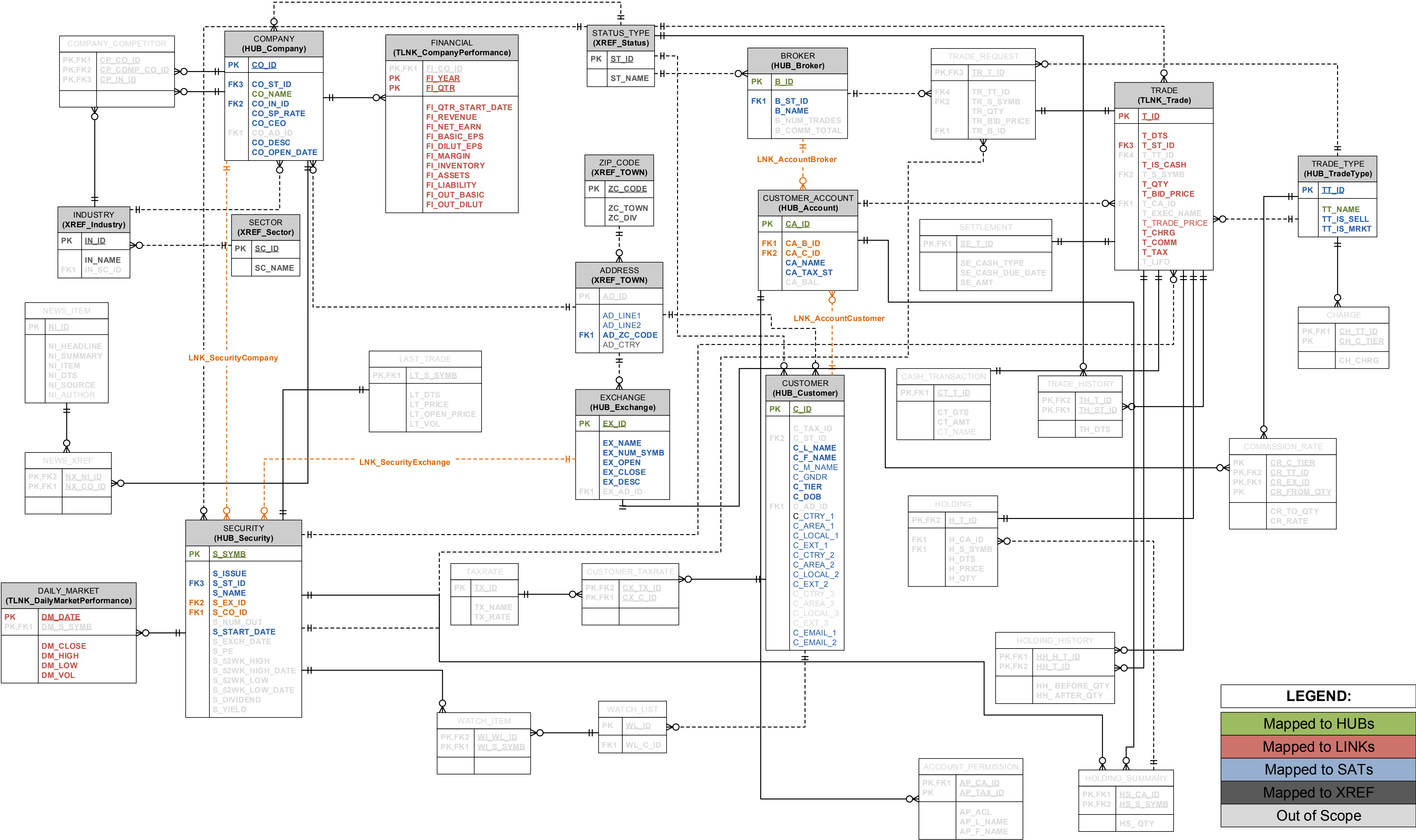
The insert below is a larger version of the TPC-E data vault model ERD as shown in chapter 4.





Appendix I - Data Vault Annotated Source Model

The insert below is a larger version of the data vault annotated source model ERD. The model colour codes different fields to show the mapping of fields between the source and target models.



# Appendix J - Data Vault Source-to-Target Mapping

This appendix contains the source-to-target mapping that was used to implement the data vault model. Table 1 to table 7 below shows the source-to-target mappings for all the hub entities while table 8 to table 11 below shows the source-to-target mappings for all the link entities. Table 12 to table 26 shows the source-to-target mappings for all the satellite entities.

Table 27 to table 29 below shows the source-to-target mappings for all the transactional link entities.



table 30 to

table 33 shows the source-to-target mappings for all the cross-reference entities.

**Table 1 Account hub (HUB\_Account) source-to-target mapping**

| Target           |              |             | Source        |                  |       |              |             | Transformation                  |
|------------------|--------------|-------------|---------------|------------------|-------|--------------|-------------|---------------------------------|
| Field            | Data Type    | Allow Nulls | Source System | Table            | Field | Data Type    | Allow Nulls |                                 |
| HUB_Account_SQN  | INT          | N           | Derived       |                  |       | INT          | N           | IDENTITY(1,1)                   |
| HUB_Account_LDTS | DATETIME2(0) | N           | Derived       |                  |       | DATETIME2(0) | N           | GETDATE when record is inserted |
| HUB_Account_RSRC | VARCHAR(10)  | N           | Derived       |                  |       | VARCHAR(10)  | N           | 'TPCE'                          |
| AccountID        | BIGINT       | N           | TPCE          | CUSTOMER_ACCOUNT | CA_ID | BIGINT       | N           |                                 |

**Table 2 Broker hub (HUB\_Broker) source-to-target mapping**

| Target          |              |             | Source        |        |       |              |             | Transformation                  |
|-----------------|--------------|-------------|---------------|--------|-------|--------------|-------------|---------------------------------|
| Field           | Data Type    | Allow Nulls | Source System | Table  | Field | Data Type    | Allow Nulls |                                 |
| HUB_Broker_SQN  | INT          | N           | Derived       |        |       | INT          | N           | IDENTITY(1,1)                   |
| HUB_Broker_LDTS | DATETIME2(0) | N           | Derived       |        |       | DATETIME2(0) | N           | GETDATE when record is inserted |
| HUB_Broker_RSRC | VARCHAR(10)  | N           | Derived       |        |       | VARCHAR(10)  | N           | 'TPCE'                          |
| BrokerID        | BIGINT       | N           | TPCE          | BROKER | B_ID  | BIGINT       | N           |                                 |

**Table 3 Company hub (HUB\_Company) source-to-target mapping**

| Target           |              |             | Source        |         |         |              |             | Transformation                  |
|------------------|--------------|-------------|---------------|---------|---------|--------------|-------------|---------------------------------|
| Field            | Data Type    | Allow Nulls | Source System | Table   | Field   | Data Type    | Allow Nulls |                                 |
| HUB_Company_SQN  | INT          | N           | Derived       |         |         | INT          | N           | IDENTITY(1,1)                   |
| HUB_Company_LDTS | DATETIME2(0) | N           | Derived       |         |         | DATETIME2(0) | N           | GETDATE when record is inserted |
| HUB_Company_RSRC | VARCHAR(10)  | N           | Derived       |         |         | VARCHAR(10)  | N           | 'TPCE'                          |
| CompanyName      | VARCHAR(60)  | N           | TPCE          | COMPANY | CO_NAME | VARCHAR(60)  | N           |                                 |

**Table 4 Customer hub (HUB\_Customer) source-to-target mapping**

| Target            |              |             | Source        |          |       |              |             | Transformation                  |
|-------------------|--------------|-------------|---------------|----------|-------|--------------|-------------|---------------------------------|
| Field             | Data Type    | Allow Nulls | Source System | Table    | Field | Data Type    | Allow Nulls |                                 |
| HUB_Customer_SQN  | INT          | N           | Derived       |          |       | INT          | N           | IDENTITY(1,1)                   |
| HUB_Customer_LDTS | DATETIME2(0) | N           | Derived       |          |       | DATETIME2(0) | N           | GETDATE when record is inserted |
| HUB_Customer_RSRC | VARCHAR(10)  | N           | Derived       |          |       | VARCHAR(10)  | N           | 'TPCE'                          |
| CustomerID        | BIGINT       | N           | TPCE          | CUSTOMER | C_ID  | BIGINT       | N           |                                 |

**Table 5 Exchange hub (HUB\_Exchange) source-to-target mapping**

| Target            |              |             | Source        |          |       |              |             | Transformation                  |
|-------------------|--------------|-------------|---------------|----------|-------|--------------|-------------|---------------------------------|
| Field             | Data Type    | Allow Nulls | Source System | Table    | Field | Data Type    | Allow Nulls |                                 |
| HUB_Exchange_SQN  | INT          | N           | Derived       |          |       | INT          | N           | IDENTITY(1,1)                   |
| HUB_Exchange_LDTS | DATETIME2(0) | N           | Derived       |          |       | DATETIME2(0) | N           | GETDATE when record is inserted |
| HUB_Exchange_RSRC | VARCHAR(10)  | N           | Derived       |          |       | VARCHAR(10)  | N           | 'TPCE'                          |
| ExchangeID        | VARCHAR(6)   | N           | TPCE          | EXCHANGE | EX_ID | VARCHAR(6)   | N           |                                 |

**Table 6 Security hub (HUB\_Security) source-to-target mapping**

| Target            |              |             | Source        |          |        |              |             | Transformation                  |
|-------------------|--------------|-------------|---------------|----------|--------|--------------|-------------|---------------------------------|
| Field             | Data Type    | Allow Nulls | Source System | Table    | Field  | Data Type    | Allow Nulls |                                 |
| HUB_Security_SQN  | INT          | N           | Derived       |          |        | INT          | N           | IDENTITY(1,1)                   |
| HUB_Security_LDTS | DATETIME2(0) | N           | Derived       |          |        | DATETIME2(0) | N           | GETDATE when record is inserted |
| HUB_Security_RSRC | VARCHAR(10)  | N           | Derived       |          |        | VARCHAR(10)  | N           | 'TPCE'                          |
| SecurityCode      | VARCHAR(15)  | N           | TPCE          | SECURITY | S_SYMB | VARCHAR(15)  | N           |                                 |

**Table 7 Trade type hub (HUB\_TradeType) source-to-target mapping**

| Target             |              |             | Source        |            |         |              |             | Transformation                  |
|--------------------|--------------|-------------|---------------|------------|---------|--------------|-------------|---------------------------------|
| Field              | Data Type    | Allow Nulls | Source System | Table      | Field   | Data Type    | Allow Nulls |                                 |
| HUB_TradeType_SQN  | INT          | N           | Derived       |            |         | INT          | N           | IDENTITY(1,1)                   |
| HUB_TradeType_LDTS | DATETIME2(0) | N           | Derived       |            |         | DATETIME2(0) | N           | GETDATE when record is inserted |
| HUB_TradeType_RSRC | VARCHAR(10)  | N           | Derived       |            |         | VARCHAR(10)  | N           | 'TPCE'                          |
| TradeTypeDesc      | VARCHAR(12)  | N           | TPCE          | TRADE_TYPE | TT_NAME | VARCHAR(12)  | N           |                                 |

**Table 8 Account broker link (LNK\_AccountBroker) source-to-target mapping**

| Target                 |              |             | Source        |             |                 |              |             | Transformation                  |
|------------------------|--------------|-------------|---------------|-------------|-----------------|--------------|-------------|---------------------------------|
| Field                  | Data Type    | Allow Nulls | Source System | Table       | Field           | Data Type    | Allow Nulls |                                 |
| LNK_AccountBroker_SQN  | INT          | N           | Derived       |             |                 | INT          | N           | IDENTITY(1,1)                   |
| LNK_AccountBroker_LDTS | DATETIME2(0) | N           | Derived       |             |                 | DATETIME2(0) | N           | GETDATE when record is inserted |
| LNK_AccountBroker_RSRC | VARCHAR(10)  | N           | Derived       |             |                 | VARCHAR(10)  | N           | 'TPCE'                          |
| HUB_Account_SQN        | INT          | N           | TPCE_DV       | HUB_Account | HUB_Account_SQN | INT          | N           |                                 |
| HUB_Broker_SQN         | INT          | N           | TPCE_DV       | HUB_Broker  | HUB_Broker_SQN  | INT          | N           |                                 |

**Table 9 Account Customer link (LNK\_AccountCustomer) source-to-target mapping**

| Target                   |              |             | Source        |              |                  |              |             | Transformation                  |
|--------------------------|--------------|-------------|---------------|--------------|------------------|--------------|-------------|---------------------------------|
| Field                    | Data Type    | Allow Nulls | Source System | Table        | Field            | Data Type    | Allow Nulls |                                 |
| LNK_AccountCustomer_SQN  | INT          | N           | Derived       |              |                  | INT          | N           | IDENTITY(1,1)                   |
| LNK_AccountCustomer_LDTS | DATETIME2(0) | N           | Derived       |              |                  | DATETIME2(0) | N           | GETDATE when record is inserted |
| LNK_AccountCustomer_RSRC | VARCHAR(10)  | N           | Derived       |              |                  | VARCHAR(10)  | N           | 'TPCE'                          |
| HUB_Account_SQN          | INT          | N           | TPCE_DV       | HUB_Account  | HUB_Account_SQN  | INT          | N           |                                 |
| HUB_Customer_SQN         | INT          | N           | TPCE_DV       | HUB_Customer | HUB_Customer_SQN | INT          | N           |                                 |

**Table 10 Security company link (LNK\_SecurityCompany) source-to-target mapping**

| Target                   |              |             | Source        |              |                  |              |             | Transformation                  |
|--------------------------|--------------|-------------|---------------|--------------|------------------|--------------|-------------|---------------------------------|
| Field                    | Data Type    | Allow Nulls | Source System | Table        | Field            | Data Type    | Allow Nulls |                                 |
| LNK_SecurityCompany_SQN  | INT          | N           | Derived       |              |                  | INT          | N           | IDENTITY(1,1)                   |
| LNK_SecurityCompany_LDTS | DATETIME2(0) | N           | Derived       |              |                  | DATETIME2(0) | N           | GETDATE when record is inserted |
| LNK_SecurityCompany_RSRC | VARCHAR(10)  | N           | Derived       |              |                  | VARCHAR(10)  | N           | 'TPCE'                          |
| HUB_Security_SQN         | INT          | N           | TPCE_DV       | HUB_Security | HUB_Security_SQN | INT          | N           |                                 |
| HUB_Company_SQN          | INT          | N           | TPCE_DV       | HUB_Company  | HUB_Company_SQN  | INT          | N           |                                 |

**Table 11 Security exchange link (LNK\_SecurityExchange) source-to-target mapping**

| Target                    |              |             | Source        |              |                  |              |             | Transformation                  |
|---------------------------|--------------|-------------|---------------|--------------|------------------|--------------|-------------|---------------------------------|
| Field                     | Data Type    | Allow Nulls | Source System | Table        | Field            | Data Type    | Allow Nulls |                                 |
| LNK_SecurityExchange_SQN  | INT          | N           | Derived       |              |                  | INT          | N           | IDENTITY(1,1)                   |
| LNK_SecurityExchange_LDTS | DATETIME2(0) | N           | Derived       |              |                  | DATETIME2(0) | N           | GETDATE when record is inserted |
| LNK_SecurityExchange_RSRC | VARCHAR(10)  | N           | Derived       |              |                  | VARCHAR(10)  | N           | 'TPCE'                          |
| HUB_Security_SQN          | INT          | N           | TPCE_DV       | HUB_Security | HUB_Security_SQN | INT          | N           |                                 |
| HUB_Exchange_SQN          | INT          | N           | TPCE_DV       | HUB_Exchange | HUB_Exchange_SQN | INT          | N           |                                 |

**Table 12 Account satellite (SAT\_Account) source-to-target mapping**

| Target            |              |             | Source        |                  |                 |              |             | Transformation                                                      |
|-------------------|--------------|-------------|---------------|------------------|-----------------|--------------|-------------|---------------------------------------------------------------------|
| Field             | Data Type    | Allow Nulls | Source System | Table            | Field           | Data Type    | Allow Nulls |                                                                     |
| HUB_Account_SQN   | INT          | N           | TPCE_DV       | HUB_Account      | HUB_Account_SQN | INT          | N           |                                                                     |
| SAT_Account_LDTS  | DATETIME2(0) | N           | Derived       |                  |                 | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_Account_LEDTS | DATETIME2(0) | Y           | Derived       |                  |                 | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_Account_RSRC  | VARCHAR(10)  | N           | Derived       |                  |                 | VARCHAR(10)  | N           | 'TPCE'                                                              |
| AccountName       | VARCHAR(50)  | N           | TPCE          | CUSTOMER_ACCOUNT | CA_NAME         | VARCHAR(50)  | N           |                                                                     |
| TaxStatusCode     | TINYINT      | N           | TPCE          | CUSTOMER_ACCOUNT | CA_TAX_ST       | TINYINT      | N           |                                                                     |

**Table 13 Account broker status satellite (SAT\_AccountBrokerStatus) source-to-target mapping**

| Target                        |              |             | Source        |                   |                       |              |             | Transformation                                                      |
|-------------------------------|--------------|-------------|---------------|-------------------|-----------------------|--------------|-------------|---------------------------------------------------------------------|
| Field                         | Data Type    | Allow Nulls | Source System | Table             | Field                 | Data Type    | Allow Nulls |                                                                     |
| LNK_AccountBroker_SQN         | INT          | N           | TPCE_DV       | LNK_AccountBroker | LNK_AccountBroker_SQN | INT          | N           |                                                                     |
| SAT_AccountBrokerStatus_LDTS  | DATETIME2(0) | N           | Derived       |                   |                       | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_AccountBrokerStatus_LEDTS | DATETIME2(0) | Y           | Derived       |                   |                       | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_AccountBrokerStatus_RSRC  | VARCHAR(10)  | N           | Derived       |                   |                       | VARCHAR(10)  | N           | 'TPCE'                                                              |

**Table 14 Account customer status satellite (SAT\_AccountCustomerStatus) source-to-target mapping**

| Target                          |              |             | Source        |                     |                         |              |             | Transformation                                                      |
|---------------------------------|--------------|-------------|---------------|---------------------|-------------------------|--------------|-------------|---------------------------------------------------------------------|
| Field                           | Data Type    | Allow Nulls | Source System | Table               | Field                   | Data Type    | Allow Nulls |                                                                     |
| LNK_AccountCustomer_SQN         | INT          | N           | TPCE_DV       | LNK_AccountCustomer | LNK_AccountCustomer_SQN | INT          | N           |                                                                     |
| SAT_AccountCustomerStatus_LDTS  | DATETIME2(0) | N           | Derived       |                     |                         | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_AccountCustomerStatus_LEDTS | DATETIME2(0) | Y           | Derived       |                     |                         | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_AccountCustomerStatus_RSRC  | VARCHAR(10)  | N           | Derived       |                     |                         | VARCHAR(10)  | N           | 'TPCE'                                                              |

**Table 15 Broker satellite (SAT\_Broker) source-to-target mapping**

| Target           |              |             | Source        |            |                |              |             | Transformation                                                      |
|------------------|--------------|-------------|---------------|------------|----------------|--------------|-------------|---------------------------------------------------------------------|
| Field            | Data Type    | Allow Nulls | Source System | Table      | Field          | Data Type    | Allow Nulls |                                                                     |
| HUB_Broker_SQN   | INT          | N           | TPCE_DV       | HUB_Broker | HUB_Broker_SQN | INT          | N           |                                                                     |
| SAT_Broker_LDTS  | DATETIME2(0) | N           | Derived       |            |                | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_Broker_LEDTS | DATETIME2(0) | Y           | Derived       |            |                | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_Broker_RSRC  | VARCHAR(10)  | N           | Derived       |            |                | VARCHAR(10)  | N           | 'TPCE'                                                              |
| BrokerName       | VARCHAR(49)  | N           | TPCE          | BROKER     | B_NAME         | VARCHAR(49)  | N           |                                                                     |
| StatusCode       | VARCHAR(4)   | N           | TPCE          | BROKER     | B_ST_ID        | TINYINT      | N           |                                                                     |

**Table 16 Company satellite (SAT\_Company) source-to-target mapping**

| Target            |              |             | Source        |             |                 |              |             | Transformation                                                      |
|-------------------|--------------|-------------|---------------|-------------|-----------------|--------------|-------------|---------------------------------------------------------------------|
| Field             | Data Type    | Allow Nulls | Source System | Table       | Field           | Data Type    | Allow Nulls |                                                                     |
| HUB_Company_SQN   | INT          | N           | TPCE_DV       | HUB_Company | HUB_Company_SQN | INT          | N           |                                                                     |
| SAT_Company_LDTS  | DATETIME2(0) | N           | Derived       |             |                 | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_Company_LEDTS | DATETIME2(0) | Y           | Derived       |             |                 | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_Company_RSRC  | VARCHAR(10)  | N           | Derived       |             |                 | VARCHAR(10)  | N           | 'TPCE'                                                              |
| CompanyID         | BIGINT       | N           | TPCE          | COMPANY     | CO_ID           | BIGINT       | N           |                                                                     |
| CompanyDesc       | VARCHAR(150) | N           | TPCE          | COMPANY     | CO_DESC         | VARCHAR(150) | N           |                                                                     |
| CompanyOpenDate   | date         | N           | TPCE          | COMPANY     | CO_OPEN_DATE    | date         | N           |                                                                     |
| AddressLine1      | VARCHAR(80)  | N           | TPCE          | ADDRESS     | AD_LINE1        | VARCHAR(80)  | N           |                                                                     |
| AddressLine2      | VARCHAR(80)  | N           | TPCE          | ADDRESS     | AD_LINE2        | VARCHAR(80)  | N           |                                                                     |
| PostalCode        | VARCHAR(12)  | N           | TPCE          | ADDRESS     | AD_ZC_CODE      | VARCHAR(80)  | N           |                                                                     |
| IndustryCode      | VARCHAR(2)   | N           | TPCE          | INDUSTRY    | IN_ID           | VARCHAR(2)   | N           |                                                                     |
| SectorCode        | VARCHAR(2)   | N           | TPCE          | INDUSTRY    | IN_SC_ID        | VARCHAR(2)   | N           |                                                                     |
| StatusCode        | VARCHAR(4)   | N           | TPCE          | COMPANY     | CO_ST_ID        | VARCHAR(4)   | N           |                                                                     |

**Table 17 Company CEO satellite (SAT\_CompanyCEO) source-to-target mapping**

| Target               |              |             | Source        |             |                 |              |             | Transformation                                                      |
|----------------------|--------------|-------------|---------------|-------------|-----------------|--------------|-------------|---------------------------------------------------------------------|
| Field                | Data Type    | Allow Nulls | Source System | Table       | Field           | Data Type    | Allow Nulls |                                                                     |
| HUB_Company_SQN      | INT          | N           | TPCE_DV       | HUB_Company | HUB_Company_SQN | INT          | N           |                                                                     |
| SAT_CompanyCEO_LDTS  | DATETIME2(0) | N           | Derived       |             |                 | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_CompanyCEO_LEDTS | DATETIME2(0) | Y           | Derived       |             |                 | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_CompanyCEO_RSRC  | VARCHAR(10)  | N           | Derived       |             |                 | VARCHAR(10)  | N           | 'TPCE'                                                              |
| CompanyCEO           | VARCHAR(46)  | N           | TPCE          | COMPANY     | CO_CEO          | VARCHAR(46)  | N           |                                                                     |



**Table 18 Company credit rating satellite (SAT\_CompanyCreditRating) source-to-target mapping**

| Target                        |              |             | Source        |             |                 |              |             | Transformation                                                      |
|-------------------------------|--------------|-------------|---------------|-------------|-----------------|--------------|-------------|---------------------------------------------------------------------|
| Field                         | Data Type    | Allow Nulls | Source System | Table       | Field           | Data Type    | Allow Nulls |                                                                     |
| HUB_Company_SQN               | INT          | N           | TPCE_DV       | HUB_Company | HUB_Company_SQN | INT          | N           |                                                                     |
| SAT_CompanyCreditRating_LDTS  | DATETIME2(0) | N           | Derived       |             |                 | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_CompanyCreditRating_LEDTS | DATETIME2(0) | Y           | Derived       |             |                 | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_CompanyCreditRating_RSRC  | VARCHAR(10)  | N           | Derived       |             |                 | VARCHAR(10)  | N           | 'TPCE'                                                              |
| CompanyCreditRating           | VARCHAR(4)   | N           | TPCE          | COMPANY     | CO_SP_RATE      | VARCHAR(4)   | N           |                                                                     |

**Table 19 Customer satellite (SAT\_Customer) source-to-target mapping**

| Target             |              |             | Source        |              |                  |              |             | Transformation                                                      |
|--------------------|--------------|-------------|---------------|--------------|------------------|--------------|-------------|---------------------------------------------------------------------|
| Field              | Data Type    | Allow Nulls | Source System | Table        | Field            | Data Type    | Allow Nulls |                                                                     |
| HUB_Customer_SQN   | INT          | N           | TPCE_DV       | HUB_Customer | HUB_Customer_SQN | INT          | N           |                                                                     |
| SAT_Customer_LDTS  | DATETIME2(0) | N           | Derived       |              |                  | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_Customer_LEDTS | DATETIME2(0) | Y           | Derived       |              |                  | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_Customer_RSRC  | VARCHAR(10)  | N           | Derived       |              |                  | VARCHAR(10)  | N           | 'TPCE'                                                              |
| LastName           | VARCHAR(25)  | N           | TPCE          | CUSTOMER     | C_L_NAME         | VARCHAR(25)  | N           |                                                                     |
| FirstName          | VARCHAR(20)  | Y           | TPCE          | CUSTOMER     | C_F_NAME         | VARCHAR(20)  | Y           |                                                                     |
| MiddleInitial      | VARCHAR(1)   | Y           | TPCE          | CUSTOMER     | C_M_NAME         | VARCHAR(1)   | Y           |                                                                     |
| Gender             | VARCHAR(1)   | N           | TPCE          | CUSTOMER     | C_GNDR           | VARCHAR(1)   | N           |                                                                     |
| DateOfBirth        | date         | N           | TPCE          | CUSTOMER     | C_DOB            | date         | N           |                                                                     |
| TierNumber         | TINYINT      | Y           | TPCE          | CUSTOMER     | C_TIER           | TINYINT      | Y           |                                                                     |
| StatusCode         | VARCHAR(4)   | Y           | TPCE          | CUSTOMER     | C_ST_ID          | VARCHAR(4)   | Y           |                                                                     |

**Table 20 Customer contact satellite (SAT\_CustomerContact) source-to-target mapping**

| Target                    |              |             | Source        |              |                  |              |             | Transformation                                                      |
|---------------------------|--------------|-------------|---------------|--------------|------------------|--------------|-------------|---------------------------------------------------------------------|
| Field                     | Data Type    | Allow Nulls | Source System | Table        | Field            | Data Type    | Allow Nulls |                                                                     |
| HUB_Customer_SQN          | INT          | N           | TPCE_DV       | HUB_Customer | HUB_Customer_SQN | INT          | N           |                                                                     |
| SAT_CustomerContact_LDTS  | DATETIME2(0) | N           | Derived       |              |                  | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_CustomerContact_LEDTS | DATETIME2(0) | Y           | Derived       |              |                  | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_CustomerContact_RSRC  | VARCHAR(10)  | N           | Derived       |              |                  | VARCHAR(10)  | N           | 'TPCE'                                                              |
| PhoneCountryCodePrimary   | VARCHAR(3)   | Y           | TPCE          | CUSTOMER     | C_CTRY_1         | VARCHAR(3)   | Y           |                                                                     |
| PhoneAreaCodePrimary      | VARCHAR(3)   | Y           | TPCE          | CUSTOMER     | C_AREA_1         | VARCHAR(3)   | Y           |                                                                     |
| PhoneLocalNumberPrimary   | VARCHAR(10)  | Y           | TPCE          | CUSTOMER     | C_LOCAL_1        | VARCHAR(10)  | Y           |                                                                     |
| PhoneExtensionPrimary     | VARCHAR(5)   | Y           | TPCE          | CUSTOMER     | C_EXT_1          | VARCHAR(5)   | Y           |                                                                     |
| PhoneCountryCodeSecondary | VARCHAR(3)   | Y           | TPCE          | CUSTOMER     | C_CTRY_2         | VARCHAR(3)   | Y           |                                                                     |
| PhoneAreaCodeSecondary    | VARCHAR(3)   | Y           | TPCE          | CUSTOMER     | C_AREA_2         | VARCHAR(3)   | Y           |                                                                     |
| PhoneLocalNumberSecondary | VARCHAR(10)  | Y           | TPCE          | CUSTOMER     | C_LOCAL_2        | VARCHAR(10)  | Y           |                                                                     |
| PhoneExtensionSecondary   | VARCHAR(5)   | Y           | TPCE          | CUSTOMER     | C_EXT_2          | VARCHAR(5)   | Y           |                                                                     |
| EmailPrimary              | VARCHAR(50)  | Y           | TPCE          | CUSTOMER     | C_EMAIL_1        | VARCHAR(50)  | Y           |                                                                     |
| EmailSecondary            | VARCHAR(50)  | Y           | TPCE          | CUSTOMER     | C_EMAIL_2        | VARCHAR(50)  | Y           |                                                                     |
| AddressLine1              | VARCHAR(80)  | N           | TPCE          | ADDRESS      | AD_LINE1         | VARCHAR(80)  | N           |                                                                     |
| AddressLine2              | VARCHAR(80)  | Y           | TPCE          | ADDRESS      | AD_LINE2         | VARCHAR(80)  | Y           |                                                                     |
| PostalCode                | VARCHAR(12)  | N           | TPCE          | ADDRESS      | AD_ZC_CODE       | VARCHAR(12)  | N           |                                                                     |

**Table 21 Exchange satellite (SAT\_Echange) source-to-target mapping**

| Target             |              |             | Source        |              |                  |              |             | Transformation                                                      |
|--------------------|--------------|-------------|---------------|--------------|------------------|--------------|-------------|---------------------------------------------------------------------|
| Field              | Data Type    | Allow Nulls | Source System | Table        | Field            | Data Type    | Allow Nulls |                                                                     |
| HUB_Exchange_SQN   | INT          | N           | TPCE_DV       | HUB_Exchange | HUB_Exchange_SQN | INT          | N           |                                                                     |
| SAT_Exchange_LDTS  | DATETIME2(0) | N           | Derived       |              |                  | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_Exchange_LEDTS | DATETIME2(0) | Y           | Derived       |              |                  | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_Exchange_RSRC  | VARCHAR(10)  | N           | Derived       |              |                  | VARCHAR(10)  | N           | 'TPCE'                                                              |
| ExchangeName       | VARCHAR(100) | N           | TPCE          | EXCHANGE     | EX_NAME          | VARCHAR(100) | N           |                                                                     |
| ExchangeDesc       | VARCHAR(150) | N           | TPCE          | EXCHANGE     | EX_DESC          | VARCHAR(150) | N           |                                                                     |
| OpeningTime        | SMALLINT     | N           | TPCE          | EXCHANGE     | EX_OPEN          | SMALLINT     | N           |                                                                     |
| ClosingTime        | SMALLINT     | N           | TPCE          | EXCHANGE     | EX_CLOSE         | SMALLINT     | N           |                                                                     |
| AddressLine1       | VARCHAR(80)  | N           | TPCE          | ADDRESS      | AD_LINE1         | VARCHAR(80)  | N           |                                                                     |
| AddressLine2       | VARCHAR(80)  | Y           | TPCE          | ADDRESS      | AD_LINE2         | VARCHAR(80)  | Y           |                                                                     |
| PostalCode         | VARCHAR(12)  | N           | TPCE          | ADDRESS      | AD_ZC_CODE       | VARCHAR(12)  | N           |                                                                     |

**Table 22 Exchange securities satellite (SAT\_EchangeSecurities) source-to-target mapping**

| Target                       |              |             | Source        |              |                  |              |             | Transformation                                                      |
|------------------------------|--------------|-------------|---------------|--------------|------------------|--------------|-------------|---------------------------------------------------------------------|
| Field                        | Data Type    | Allow Nulls | Source System | Table        | Field            | Data Type    | Allow Nulls |                                                                     |
| HUB_Exchange_SQN             | INT          | N           | TPCE_DV       | HUB_Exchange | HUB_Exchange_SQN | INT          | N           |                                                                     |
| SAT_ExchangeSecurities_LDTS  | DATETIME2(0) | N           | Derived       |              |                  | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_ExchangeSecurities_LEDTS | DATETIME2(0) | Y           | Derived       |              |                  | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_ExchangeSecurities_RSRC  | VARCHAR(10)  | N           | Derived       |              |                  | VARCHAR(10)  | N           | 'TPCE'                                                              |
| ListedSecurities             | INT          | N           | TPCE          | EXCHANGE     | EX_NUM_SYMB      | INT          | N           |                                                                     |

**Table 23 Security satellite (SAT\_Security) source-to-target mapping**

| Target             |              |             | Source        |              |                  |              |             | Transformation                                                      |
|--------------------|--------------|-------------|---------------|--------------|------------------|--------------|-------------|---------------------------------------------------------------------|
| Field              | Data Type    | Allow Nulls | Source System | Table        | Field            | Data Type    | Allow Nulls |                                                                     |
| HUB_Security_SQN   | INT          | N           | TPCE_DV       | HUB_Security | HUB_Security_SQN | INT          | N           |                                                                     |
| SAT_Security_LDTS  | DATETIME2(0) | N           | Derived       |              |                  | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_Security_LEDTS | DATETIME2(0) | Y           | Derived       |              |                  | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_Security_RSRC  | VARCHAR(10)  | N           | Derived       |              |                  | VARCHAR(10)  | N           | 'TPCE'                                                              |
| SecurityName       | VARCHAR(70)  | N           | TPCE          | EXCHANGE     | S_NAME           | VARCHAR(70)  | N           |                                                                     |
| IssueType          | VARCHAR(6)   | N           | TPCE          | SECURITY     | S_ISSUE          | VARCHAR(6)   | N           |                                                                     |
| FirstTradeDate     | date         | N           | TPCE          | SECURITY     | S_START_DATE     | date         | N           |                                                                     |
| StatusCode         | VARCHAR(4)   | N           | TPCE          | SECURITY     | S_ST_ID          | VARCHAR(4)   | N           |                                                                     |

**Table 24 Security company status satellite (SAT\_SecurityCompanyStatus) source-to-target mapping**

| Target                          |              |             | Source        |                     |                         |              |             | Transformation                                                      |
|---------------------------------|--------------|-------------|---------------|---------------------|-------------------------|--------------|-------------|---------------------------------------------------------------------|
| Field                           | Data Type    | Allow Nulls | Source System | Table               | Field                   | Data Type    | Allow Nulls |                                                                     |
| LNK_SecurityCompany_SQN         | INT          | N           | TPCE_DV       | LNK_SecurityCompany | LNK_SecurityCompany_SQN | INT          | N           |                                                                     |
| SAT_SecurityCompanyStatus_LDTS  | DATETIME2(0) | N           | Derived       |                     |                         | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_SecurityCompanyStatus_LEDTS | DATETIME2(0) | Y           | Derived       |                     |                         | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_SecurityCompanyStatus_RSRC  | VARCHAR(10)  | N           | Derived       |                     |                         | VARCHAR(10)  | N           | 'TPCE'                                                              |

**Table 25 Security exchange status satellite (SAT\_SecurityExchangeStatus) source-to-target mapping**

| Target                           |              |             | Source        |                      |                          |              |             | Transformation                                                      |
|----------------------------------|--------------|-------------|---------------|----------------------|--------------------------|--------------|-------------|---------------------------------------------------------------------|
| Field                            | Data Type    | Allow Nulls | Source System | Table                | Field                    | Data Type    | Allow Nulls |                                                                     |
| LNK_SecurityExchange_SQN         | INT          | N           | TPCE_DV       | LNK_SecurityExchange | LNK_SecurityExchange_SQN | INT          | N           |                                                                     |
| SAT_SecurityExchangeStatus_LDTS  | DATETIME2(0) | N           | Derived       |                      |                          | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_SecurityExchangeStatus_LEDTS | DATETIME2(0) | Y           | Derived       |                      |                          | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_SecurityExchangeStatus_RSRC  | VARCHAR(10)  | N           | Derived       |                      |                          | VARCHAR(10)  | N           | 'TPCE'                                                              |

**Table 26 Trade type satellite (SAT\_TradeType) source-to-target mapping**

| Target              |              |             | Source        |               |                   |              |             | Transformation                                                      |
|---------------------|--------------|-------------|---------------|---------------|-------------------|--------------|-------------|---------------------------------------------------------------------|
| Field               | Data Type    | Allow Nulls | Source System | Table         | Field             | Data Type    | Allow Nulls |                                                                     |
| HUB_TradeType_SQN   | INT          | N           | TPCE_DV       | HUB_TradeType | HUB_TradeType_SQN | INT          | N           |                                                                     |
| SAT_TradeType_LDTS  | DATETIME2(0) | N           | Derived       |               |                   | DATETIME2(0) | N           | GETDATE when record is inserted                                     |
| SAT_TradeType_LEDTS | DATETIME2(0) | Y           | Derived       |               |                   | DATETIME2(0) | Y           | NULL until record is end-dated.<br>GETDATE when record is end-dated |
| SAT_TradeType_RSRC  | VARCHAR(10)  | N           | Derived       |               |                   | VARCHAR(10)  | N           | 'TPCE'                                                              |
| TradeTypeCode       | VARCHAR(3)   | N           | TPCE          | TRADE_TYPE    | TT_ID             | VARCHAR(3)   | N           |                                                                     |
| IsSellFlag          | bit          | N           | TPCE          | TRADE_TYPE    | TT_IS_SELL        | bit          | N           |                                                                     |
| IsMarketFlag        | bit          | N           | TPCE          | TRADE_TYPE    | TT_IS_MRKT        | bit          | N           |                                                                     |

**Table 27 Daily market performance transactional link (TLNK\_DailyMarketPerformance) source-to-target mapping**

| Target                           |              |             | Source        |              |                  |              |             | Transformation                  |
|----------------------------------|--------------|-------------|---------------|--------------|------------------|--------------|-------------|---------------------------------|
| Field                            | Data Type    | Allow Nulls | Source System | Table        | Field            | Data Type    | Allow Nulls |                                 |
| HUB_Security_SQN                 | INT          | N           | TPCE_DV       | HUB_Security | HUB_Security_SQN | INT          | N           |                                 |
| Date                             | date         | N           | TPCE          | DAILY_MARKET | DM_DATE          | DATE         | N           |                                 |
| TLNK_DailyMarketPerformance_LDTS | DATETIME2(0) | N           | Derived       |              |                  | DATETIME2(0) | Y           | GETDATE when record is inserted |
| TLNK_DailyMarketPerformance_RSRC | VARCHAR(10)  | N           | Derived       |              |                  | VARCHAR(10)  | N           | 'TPCE'                          |
| HUB_Company_SQN                  | INT          | N           | TPCE_DV       | HUB_Company  | HUB_Company_SQN  | INT          | N           |                                 |
| HUB_Exchange_SQN                 | INT          | N           | TPCE_DV       | HUB_Exchange | HUB_Exchange_SQN | INT          | N           |                                 |
| DayClose                         | money        | N           | TPCE          | DAILY_MARKET | DM_CLOSE         | money        | N           |                                 |
| DayHigh                          | money        | N           | TPCE          | DAILY_MARKET | DM_HIGH          | money        | N           |                                 |
| DayLow                           | money        | N           | TPCE          | DAILY_MARKET | DM_LOW           | money        | N           |                                 |
| DayVolume                        | BIGINT       | N           | TPCE          | DAILY_MARKET | DM_VOL           | BIGINT       | N           |                                 |

**Table 28 Quarterly company performance transactional link (TLNK\_QuarterlyCompanyPerformance) source-to-target mapping**

| Target                                |              |             | Source        |             |                   |              |             | Transformation                  |
|---------------------------------------|--------------|-------------|---------------|-------------|-------------------|--------------|-------------|---------------------------------|
| Field                                 | Data Type    | Allow Nulls | Source System | Table       | Field             | Data Type    | Allow Nulls |                                 |
| HUB_Company_SQN                       | INT          | N           | TPCE_DV       | HUB_Company | HUB_Company_SQN   | INT          | N           |                                 |
| FiscalQuarter                         | INT          | N           | TPCE          | FINANCIAL   | FI_QTR            | INT          | N           |                                 |
| FiscalYear                            | INT          | N           | TPCE          | FINANCIAL   | FI_YEAR           | INT          | N           |                                 |
| TLNK_QuarterlyCompanyPerformance_LDTS | DATETIME2(0) | N           | Derived       | FINANCIAL   |                   | DATETIME2(0) | Y           | GETDATE when record is inserted |
| TLNK_QuarterlyCompanyPerformance_RSRC | VARCHAR(10)  | N           | Derived       | FINANCIAL   |                   | VARCHAR(10)  | N           | 'TPCE'                          |
| FiscalQuarterStartDate                | date         | N           | TPCE          | FINANCIAL   | FI_QTR_START_DATE | date         | N           |                                 |
| Revenue                               | money        | N           | TPCE          | FINANCIAL   | FI_REVENUE        | money        | N           |                                 |
| NetEarnings                           | money        | N           | TPCE          | FINANCIAL   | FI_NET_EARN       | money        | N           |                                 |
| EarningsPerShareBasic                 | money        | N           | TPCE          | FINANCIAL   | FI_BASIC_EPS      | money        | N           |                                 |
| EarningsPerShareDiluted               | money        | N           | TPCE          | FINANCIAL   | FI_DILUT_EPS      | money        | N           |                                 |
| Margin                                | money        | N           | TPCE          | FINANCIAL   | FI_MARGIN         | money        | N           |                                 |
| Inventory                             | money        | N           | TPCE          | FINANCIAL   | FI_INVENTORY      | money        | N           |                                 |
| Assets                                | money        | N           | TPCE          | FINANCIAL   | FI_ASSETS         | money        | N           |                                 |
| Liabilities                           | money        | N           | TPCE          | FINANCIAL   | FI_LIABILITY      | money        | N           |                                 |
| SharesOutstandingBasicAvg             | BIGINT       | N           | TPCE          | FINANCIAL   | FI_OUT_BASIC      | BIGINT       | N           |                                 |
| SharesOutstandingDilutedAvg           | BIGINT       | N           | TPCE          | FINANCIAL   | FI_OUT_DILUT      | BIGINT       | N           |                                 |

**Table 29 Trade transactional link (TLNK\_Trade) source-to-target mapping**

| Target            |              |             | Source        |               |                   |              |             | Transformation                  |
|-------------------|--------------|-------------|---------------|---------------|-------------------|--------------|-------------|---------------------------------|
| Field             | Data Type    | Allow Nulls | Source System | Table         | Field             | Data Type    | Allow Nulls |                                 |
| TradeID           | BIGINT       | N           | TPCE          | TRADE         | T_ID              | BIGINT       | N           |                                 |
| TLNK_Trade_LDTS   | DATETIME2(0) | N           | Derived       |               |                   | DATETIME2(0) | Y           | GETDATE when record is inserted |
| TLNK_Trade_RSRC   | VARCHAR(10)  | N           | Derived       |               |                   | VARCHAR(10)  | N           | 'TPCE'                          |
| HUB_Broker_SQN    | INT          | N           | TPCE_DV       | HUB_Broker    | HUB_Broker_SQN    | INT          | N           |                                 |
| HUB_Customer_SQN  | INT          | N           | TPCE_DV       | HUB_Customer  | HUB_Customer_SQN  | INT          | N           |                                 |
| HUB_Account_SQN   | INT          | N           | TPCE_DV       | HUB_Account   | HUB_Account_SQN   | INT          | N           |                                 |
| HUB_Company_SQN   | INT          | N           | TPCE_DV       | HUB_Company   | HUB_Company_SQN   | INT          | N           |                                 |
| HUB_Exchange_SQN  | INT          | N           | TPCE_DV       | HUB_Exchange  | HUB_Exchange_SQN  | INT          | N           |                                 |
| HUB_Security_SQN  | INT          | N           | TPCE_DV       | HUB_Security  | HUB_Security_SQN  | INT          | N           |                                 |
| HUB_TradeType_SQN | INT          | N           | TPCE_DV       | HUB_TradeType | HUB_TradeType_SQN | INT          | N           |                                 |
| TradeDATETIME     | DATETIME2(0) | N           | TPCE          | TRADE         | T_DTS             | DATETIME2(0) | N           |                                 |
| Quantity          | INT          | N           | TPCE          | TRADE         | T_QTY             | INT          | N           |                                 |
| BidPrice          | money        | N           | TPCE          | TRADE         | T_BID_PRICE       | money        | N           |                                 |
| TradePrice        | money        | N           | TPCE          | TRADE         | T_TRADE_PRICE     | money        | N           |                                 |
| Charge            | money        | N           | TPCE          | TRADE         | T_CHRG            | money        | N           |                                 |
| Commission        | money        | N           | TPCE          | TRADE         | T_COMM            | money        | N           |                                 |
| Tax               | money        | N           | TPCE          | TRADE         | T_TAX             | money        | N           |                                 |
| IsCashFlag        | bit          | N           | TPCE          | TRADE         | T_IS_CASH         | bit          | N           |                                 |
| StatusCode        | VARCHAR(4)   | N           | TPCE          | TRADE         | T_ST_ID           | VARCHAR(4)   | N           |                                 |

**Table 30 Industry reference (XREF\_Industry) source-to-target mapping**

| Target             |              |             | Source        |          |         |              |             | Transformation                  |
|--------------------|--------------|-------------|---------------|----------|---------|--------------|-------------|---------------------------------|
| Field              | Data Type    | Allow Nulls | Source System | Table    | Field   | Data Type    | Allow Nulls |                                 |
| IndustryCode       | VARCHAR(2)   | N           | TPCE          | INDUSTRY | IN_ID   | VARCHAR(2)   | N           |                                 |
| XREF_Industry_LDTS | DATETIME2(0) | N           | Derived       |          |         | DATETIME2(0) | N           | GETDATE when record is inserted |
| XREF_Industry_RSRC | VARCHAR(10)  | N           | Derived       |          |         | VARCHAR(10)  | N           | 'TPCE'                          |
| IndustryDesc       | VARCHAR(50)  | N           | TPCE          | INDUSTRY | IN_NAME | VARCHAR(50)  | N           |                                 |

**Table 31 Sector reference (XREF\_Sector) source-to-target mapping**

| Target           |              |             | Source        |        |         |              |             | Transformation                  |
|------------------|--------------|-------------|---------------|--------|---------|--------------|-------------|---------------------------------|
| Field            | Data Type    | Allow Nulls | Source System | Table  | Field   | Data Type    | Allow Nulls |                                 |
| SectorCode       | VARCHAR(2)   | N           | TPCE          | SECTOR | SC_ID   | VARCHAR(2)   | N           |                                 |
| XREF_Sector_LDTS | DATETIME2(0) | N           | Derived       |        |         | DATETIME2(0) | N           | GETDATE when record is inserted |
| XREF_Sector_RSRC | VARCHAR(10)  | N           | Derived       |        |         | VARCHAR(10)  | N           | 'TPCE'                          |
| SectorDesc       | VARCHAR(30)  | N           | TPCE          | SECTOR | SC_NAME | VARCHAR(30)  | N           |                                 |

**Table 32 Status reference (XREF\_Status) source-to-target mapping**

| Target           |              |             | Source        |             |         |              |             | Transformation                  |
|------------------|--------------|-------------|---------------|-------------|---------|--------------|-------------|---------------------------------|
| Field            | Data Type    | Allow Nulls | Source System | Table       | Field   | Data Type    | Allow Nulls |                                 |
| StatusCode       | VARCHAR(4)   | N           | TPCE          | STATUS_TYPE | ST_ID   | VARCHAR(4)   | N           |                                 |
| XREF_Status_LDTS | DATETIME2(0) | N           | Derived       |             |         | DATETIME2(0) | N           | GETDATE when record is inserted |
| XREF_Status_RSRC | VARCHAR(10)  | N           | Derived       |             |         | VARCHAR(10)  | N           | 'TPCE'                          |
| StatusDesc       | VARCHAR(10)  | N           | TPCE          | STATUS_TYPE | ST_NAME | VARCHAR(10)  | N           |                                 |



**Table 33 Town reference (XREF\_Town) source-to-target mapping**

| Target            |              |             | Source        |          |         |              |             | Transformation                  |
|-------------------|--------------|-------------|---------------|----------|---------|--------------|-------------|---------------------------------|
| Field             | Data Type    | Allow Nulls | Source System | Table    | Field   | Data Type    | Allow Nulls |                                 |
| PostalCode        | VARCHAR(12)  | N           | TPCE          | ZIP_CODE | ST_ID   | VARCHAR(4)   | N           |                                 |
| XREF_Town_LDTS    | DATETIME2(0) | N           | Derived       |          |         | DATETIME2(0) | N           | GETDATE when record is inserted |
| XREF_Town_RSRC    | VARCHAR(10)  | N           | Derived       |          |         | VARCHAR(10)  | N           | 'TPCE'                          |
| TownName          | VARCHAR(80)  | N           | TPCE          | ZIP_CODE | ST_NAME | VARCHAR(10)  | N           |                                 |
| StateProvinceCode | VARCHAR(80)  | N           | TPCE          | ZIP_CODE | ZC_DIV  | VARCHAR(80)  | N           |                                 |
| CountryName       | VARCHAR(80)  | N           | TPCE          | ADDRESS  | AD_CTRY | VARCHAR(80)  | N           |                                 |

# **Appendix K - Martijn Evers' Data Vault Model Review**

Below is a copy of the data vault model review report from Martijn Evers, a recognised expert in the field of business Intelligence and data vault modelling. Evers' review was based on the following three artefacts:

- Annotated TPC-E ERD (Appendix I - Data Vault Annotated Source Model)
- Data vault source-to-target mapping (Appendix J - Data Vault Source-to-Target Mapping)
- Data vault model ERD (Appendix H - Full Sized TPC-E Data Vault Model)

This report contains an evaluation of the Data Vault created by Marius van Schalkwyk by Dutch Data Vault expert Martijn Evers.

## **A. RELEVANT ACADEMIC QUALIFICATIONS**

Martijn Evers has earned a Doctor's degree in Computer Science (comparable to a M.Sc.) at the Dutch Radboud University of Nijmegen.

## **B. DATA VAULT QUALIFICATIONS**

Martijn Evers is considered by many in the Dutch BI & Data Vault community as one of the principal experts on data vault modeling and other temporal modeling techniques. He is a data warehouse architect at the Radboud University of Nijmegen, President of the Dutch Data Vault user group and member of the Dutch section of the International Data Vault Standards board. He was awarded Dutch "BI Consultant of the Year" 2011 as recognition of his commitment to the Dutch BI & Data Vault community. Martijn is a data vault auditor and he also works as external architect and consultant on several data vault projects. He is also a Data Vault instructor and mentor. He does academic research, tutoring and M.Sc. program support for the University of Applied Science, Arnhem & Nijmegen (HAN), for their BI Master program.

## **C. EVALUATION OF THE DATA VAULT**

The artifacts that have been evaluated are an annotated source model, a Data Vault model (both in MS Visio) and a column/key mapping made in Microsoft Excel. Based on these artifacts and the input of its creator I conclude that this data vault model conforms to the data vault standards as laid down by its creator, Dan Linstedt.

# Appendix L - Load performance experiment protocol

## 1.1 Purpose

The purpose of this experiment is to compare the time it takes to load data from a source system into data warehouses based on a dimensional data model and a data vault model respectively.

The hypothesis for this experiment is formulated as follow:

- The average time required to load historical data into a data vault model is less than the average time required to load the same data into a dimensional model.
- The average time required to load incremental data into a data vault model is less than the average time required to load the same data into a dimensional model.

## 1.2 Equipment

- A testing server with the following software:
  - Windows Server 2008 R2 Enterprise Edition Service Pack 1.
  - SQL Server 2008 Service Pack 2.\

The specification and configuration for the server is given in *appendix O - Server Specification*, *appendix P - SQL Server Configuration* and *appendix A - Disabled Windows Services*.

- Historical TPC-E dataset loaded in a SQL Server database.
- 30 sets of incremental data in the source database that can be used for the incremental load.

## 1.3 Preparation

1. Install the operating system on the testing server. Configure the operating system for optimal performance. Disable as many processes/services on the testing server as is possible in order to reduce the probability of other processes/services influencing the results. Exact details of the server optimisation can be found in *Appendix A - Disabled Windows Services*.

2. Install SQL Server on the testing server. Configure SQL Server for optimal performance. A reference of this configuration is detailed in *Appendix P - SQL Server Configuration*.
3. Create TPC-E source database including all objects (tables, views, indexes, etc.). The t-SQL script that was used to create these objects is detailed in *Appendix Q - SQL Server T-SQL Scripts, 1.1.1 TPCE Create Script*.
4. Generate historical TPC-E dataset. Details of the data generation process are detailed in *Appendix B - Sample Data Generation*.
5. Import historical TPC-E dataset into TPCE source database from the source files. Details of how the historical data files were processed are detailed in *Appendix Q - SQL Server T-SQL Scripts, 1.2.1.1 TPCE Historical Data Import T-SQL Script*.
6. Generate incremental TPC-E dataset. Details of the data generation process are detailed in *Appendix B - Sample Data Generation*. The T-SQL script that was used to import the data is detailed in *Appendix Q - SQL Server T-SQL Scripts, 1.2.2.1 TPC Incremental Data Import T-SQL Script*.
7. Design dimensional and data vault models (Target data models). The dimensional model was designed using the four-step method as described by Kimball *et al.* (2008:246). The resulting dimensional model is detailed *Appendix E - Full Sized TPC-E Dimensional Model*. The data vault model was designed using the seven-step method described in chapter 3.
8. Peer review of target models. The models were peer-reviewed by Van Zyl Kruger, whose review report can be found in *Appendix F - Van Zyl Kruger's Dimensional Model Review* and Martijn Evers, whose review report can be found in *Appendix K - Martijn Evers' Data Vault Model Review*.
9. Create source-to-target mapping (STTM) for both target models. The SSTMs for the dimensional model and the data vault model is detailed in *Appendix G - Dimensional Source-to-Target Mapping* and *Appendix J - Data Vault Source-to-Target Mapping* respectively.
10. Physical implementation of target models on DBMS. The T-SQL script that was used to create the target database and objects can be found in *Appendix Q - SQL Server T-SQL Scripts, section 1.1.2 TPCE\_DM Create Script* and *1.1.3 TPCE\_DV Create Script*.
11. Create history load extract, transform and load (ETL) queries that will perform ETL of historical data from the source system into the target databases. These T-SQL scripts can be found in *Appendix Q - SQL Server T-SQL Scripts, section 1.2.1.2 TPCE\_DM Historical Data Loading T-SQL Script* and *1.2.1.3 TPCE\_DV Historical Data Loading*.

12. Create incremental load ETL queries that will perform ETL of incremental data from the source system into the target databases. These T-SQL scripts can be found in *Appendix Q - SQL Server T-SQL Scripts*, section 1.2.2.2 *TPCE\_DM Incremental Data Loading T-SQL Script* and 1.2.2.3 *TPCE\_DV Incremental Data Loading T-SQL Script*.
13. Test history and incremental ETL queries. Ensure correctness of the above queries by using them to populating the target data warehouses with the data and then balancing data between the two warehouses using the scripts shown in *Appendix Q - SQL Server T-SQL Scripts*, section 1.3 *Data Balancing Scripts*.
14. Determine the order in which the queries for the two target models needs to be executed. The order in which the queries need to be executed in is determined by the dependencies between them. The sequence is detailed below in table 1 and table 2 for the dimensional model and data vault model respectively.

**Table 1 Dimensional model load execution sequence**

| Table                         | Sequence | History Load | Incremental Load |
|-------------------------------|----------|--------------|------------------|
| D_Broker                      | 1        | X            | X                |
| D_CashType                    | 1        | X            | X                |
| D_Company                     | 1        | X            | X                |
| D_CustomerAccount             | 1        | X            | X                |
| D_Date                        | 1        | X            |                  |
| D_Exchange                    | 1        | X            | X                |
| D_FiscalQuarter               | 2        | X            |                  |
| D_Security                    | 1        | X            | X                |
| D_TradeStatus                 | 1        | X            | X                |
| D_TradeType                   | 1        | X            | X                |
| F_DailyMarketPerformance      | 3        | X            | X                |
| F_QuarterlyCompanyPerformance | 4        | X            |                  |
| F_Trade                       | 5        | X            | X                |

**Table 2 Data vault model load execution sequence**

| Table        | Sequence | History Load | Incremental Load |
|--------------|----------|--------------|------------------|
| HUB_Account  | 1        | X            | X                |
| HUB_Broker   | 1        | X            | X                |
| HUB_Company  | 1        | X            | X                |
| HUB_Customer | 1        | X            | X                |
| HUB_Exchange | 1        | X            | X                |
| HUB_Security | 1        | X            | X                |

|                                  |   |   |   |
|----------------------------------|---|---|---|
| HUB_TradeType                    | 1 | X | X |
| XREF_Industry                    | 1 | X | X |
| XREF_Sector                      | 1 | X | X |
| XREF_Status                      | 1 | X | X |
| XREF_Town                        | 1 | X | X |
| LNK_AccountBroker                | 2 | X | X |
| LNK_AccountCustomer              | 2 | X | X |
| LNK_SecurityCompany              | 2 | X | X |
| LNK_SecurityExchange             | 2 | X | X |
| SAT_Account                      | 2 | X | X |
| SAT_Broker                       | 2 | X | X |
| SAT_Company                      | 2 | X | X |
| SAT_CompanyCEO                   | 2 | X | X |
| SAT_CompanyCreditRating          | 2 | X | X |
| SAT_Customer                     | 2 | X | X |
| SAT_CustomerContact              | 2 | X | X |
| SAT_Exchange                     | 2 | X | X |
| SAT_ExchangeSecurities           | 2 | X | X |
| SAT_Security                     | 2 | X | X |
| SAT_TradeType                    | 2 | X | X |
| SAT_AccountBrokerStatus          | 3 | X | X |
| SAT_AccountCustomerStatus        | 3 | X | X |
| SAT_SecurityCompanyStatus        | 3 | X | X |
| SAT_SecurityExchangeStatus       | 3 | X | X |
| TLNK_DailyMarketPerformance      | 4 | X | X |
| TLNK_QuarterlyCompanyPerformance | 5 | X | X |
| TLNK_Trade                       | 6 | X | X |

## 1.4 Execution

### 1.4.1 History Load

1. For each of the dimensional model ETL queries that loads the historical data, repeat the following steps 10 times:
  - a. Truncate all the target tables (not required on first iteration as the tables are already empty).
  - b. Restart the SQL Server database service in order to clear the cache and reset the memory allocated to SQL Server.
  - c. Execute the query through SQL Server command line utility (sqlcmd.exe)
  - d. Record the duration of the query.

2. For each of the data vault model ETL queries that loads the historical data, repeat the following steps 10 times:
  - a. Truncate all the target tables (not required on first iteration as the tables are already empty).
  - b. Restart the SQL Server database service in order to clear the cache and reset the memory allocated to SQL Server.
  - c. Execute the query through SQL Server command line utility (sqlcmd.exe)
  - d. Record the duration of the query.
3. Make a full backup of the TPCE database.

#### **1.4.2 Incremental Load**

1. Restore the TPCE backup that was created in point 3 in section *1.4.1 History Load* above.
2. Load the first set of incremental data into the source system database.
3. For each of the dimensional model ETL queries that loads the incremental data:
  - a. Restart the SQL Server database service in order to clear the cache and reset the memory allocated to SQL Server.
  - b. Execute the query through SQL Server command line utility (sqlcmd.exe)
  - c. Record the duration of the query.
4. For each of the data vault model ETL queries that loads the incremental data:
  - a. Restart the SQL Server database service in order to clear the cache and reset the memory allocated to SQL Server.
  - b. Execute the query through SQL Server command line utility (sqlcmd.exe)
  - c. Record the duration of the query.
5. Repeat steps 3 and 4 for each of the thirty incremental datasets.
6. Repeat steps 1 to 5 above three times.

### **1.5 Data Analysis**

The data analysis for the experiment was done by the Statistical Consultation Services at the North-West University Potchefstroom campus who determined that the t-test for dependent samples and the Wilcoxon Matched Pairs test analysis methods is best suited to analyse the results.

The results of the analysis were interpreted by the researcher and are discussed in chapter 4.



# Appendix M - Query Performance Experiment Protocol

## 1.1 Purpose

The purpose of this experiment is to compare the time it takes to execute equivalent queries against each of the target data models.

The hypothesis for this experiment is formulated as follow:

The average query execution time for queries to a data vault model is less than the average query response time for similar queries to the dimensional Model.

## 1.2 Equipment

- A testing server with the following software:
  - Windows Server 2008 R2 Enterprise Edition Service Pack 1.
  - SQL Server 2008 Service Pack 2.

The specification and configuration for the server is given in *appendix O - Server Specification*, *appendix P - SQL Server Configuration* and *appendix A - Disabled Windows Services*.

- Historical and 30 incremental TPC-E datasets loaded into each of the target data models.

## 1.3 Preparation

1. This experiment follows on the load performance experiment. It is dependent on the execution of that experiment and the preparation steps as described in *Appendix L - Load performance experiment protocol*.
2. Create the T-SQL queries in each of the categories identified in chapter 4 from each of the target data models. The queries are shown in *Appendix Q – SQL Server T-SQL Scripts*, section *1.4.1 Dimensional Model* and *1.4.2 Data Vault Model*.
3. Validate that the respective queries from the dimensional model and the data vault model would return exactly the same results by balancing the result sets.

## **1.4 Execution**

1. For each of the dimensional model queries:
  - a. Restart the SQL Server database service in order to clear the cache and reset the memory allocated to SQL Server.
  - b. Execute the query through SQL Server command line utility (sqlcmd.exe)
  - c. Record the duration of the query.
2. For each of the data vault model queries:
  - a. Restart the SQL Server database service in order to clear the cache and reset the memory allocated to SQL Server.
  - b. Execute the query through SQL Server command line utility (sqlcmd.exe)
  - c. Record the duration of the query.
3. Repeat steps 1 and 2 above fifteen times.

## **1.5 Data Analysis**

The data analysis for the experiment was done by the Statistical Consultation Services at the North-West University Potchefstroom campus who determined that the t-test for dependent samples analysis method is best suited to analyse the results.

The results of the analysis were interpreted by the researcher and are discussed in chapter 4.

# Appendix N - Storage Size Experiment Protocol

## 1.1 Purpose

This experiment is aimed at comparing dimensional and data vault models in terms of storage space required.

The hypothesis for this experiment is formulated as follow:

The average storage space required for data in a data vault model is less than the average storage space required for the same data in a dimensional model.

## 1.2 Equipment

- A testing server with the following software:
  - Windows Server 2008 R2 Enterprise Edition Service Pack 1.
  - SQL Server 2008 Service Pack 2.

The specification and configuration for the server is given in *appendix O - Server Specification*, *appendix P - SQL Server Configuration* and *appendix A - Disabled Windows Services*.

- Historical TPC-E dataset loaded in a SQL Server database.
- 30 Sets of incremental data in the source database that can be used for the incremental loads.

## 1.3 Preparation

1. Install the operating system on the testing server. Configure the operating system for optimal performance. Disable as many processes/services on the testing server as is possible in order to reduce the probability of other processes/services influencing the results. Exact details of the server optimisation can be found in *Appendix A - Disabled Windows Services*.
2. Install SQL Server on the testing server. Configure SQL Server for optimal performance. A reference of this configuration is detailed in *Appendix P - SQL Server Configuration*.

3. Create TPC-E source database including all objects (tables, views, indexes, etc.). The t-SQL script that was used to create these objects is detailed in *Appendix Q - SQL Server T-SQL Scripts, 1.1.1 TPCE Create Script*.
4. Generate historical TPC-E dataset. Details of the data generation process are detailed in *Appendix B - Sample Data Generation*.
5. Import historical TPC-E dataset into TPCE source database from the source files. Details of how the historical data files were processed are detailed in *Appendix Q - SQL Server T-SQL Scripts, 1.2.1.1 TPCE Historical Data Import T-SQL Script*.
6. Generate incremental TPC-E dataset. Details of the data generation process are detailed in *Appendix B - Sample Data Generation*. The T-SQL script that was used to import the data is detailed in *Appendix Q - SQL Server T-SQL Scripts, 1.2.2.1 TPC Incremental Data Import T-SQL Script*.
7. Design dimensional and data vault models (Target data models). The dimensional model was designed using the four-step method as described by Kimball *et al.* (2008:246). The resulting dimensional model is detailed *Appendix E - Full Sized TPC-E Dimensional Model*. The data vault model was designed using the seven-step method described in chapter 3.
8. Peer review of target models. The models were peer-reviewed by Van Zyl Kruger, whose review report can be found in *Appendix F - Van Zyl Kruger's Dimensional Model Review* and Martijn Evers, whose review report can be found in *Appendix K - Martijn Evers' Data Vault Model Review*.
9. Create source-to-target mapping (STTM) for both target models. The SSTMs for the dimensional model and the data vault model is detailed in *Appendix G - Dimensional Source-to-Target Mapping* and *Appendix J - Data Vault Source-to-Target Mapping* respectively.
10. Physical implementation of target models on DBMS. The T-SQL script that was used to create the target database and objects can be found in *Appendix Q - SQL Server T-SQL Scripts, section 1.1.2 TPCE\_DM Create Script* and *1.1.3 TPCE\_DV Create Script*.
11. Create history load extract, transform and load (ETL) queries that will perform ETL of historical data from the source system into the target databases. These T-SQL scripts can be found in *Appendix Q - SQL Server T-SQL Scripts, section 1.2.1.2 TPCE\_DM Historical Data Loading T-SQL Script* and *1.2.1.3 TPCE\_DV Historical Data Loading*.
12. Create incremental load ETL queries that will perform ETL of incremental data from the source system into the target databases. These T-SQL scripts can be found in *Appendix Q - SQL Server T-SQL Scripts, section 1.2.2.2 TPCE\_DM Incremental*

*Data Loading T-SQL Script and 1.2.2.3 1.2.2.2 TPCE\_DV Incremental Data Loading T-SQL Script.*

13. Test history and incremental ETL queries. Ensure correctness of the above queries by using them to populating the target data warehouses with the data and then balancing data between the two warehouses using the scripts shown in *Appendix Q - SQL Server T-SQL Scripts*, section *1.3 Data Balancing Scripts*.

## **1.4 Execution**

1. Load the first set of incremental data into the source system database and subsequently into each of the target data models as described in *Appendix L - Load Performance Experiment Protocol*.
2. Record the data and index size for each of the target data models. The queries that was used to capture this information is detailed in *Appendix Q - SQL Server T-SQL Scripts*, section *1.5 Storage Size Queries*.
3. Repeat steps 1 and 2 for each of the remaining 29 incremental datasets.

## **1.5 Data Analysis**

The data analysis for the experiment was done by the Statistical Consultation Services at the North-West University Potchefstroom campus who determined that the t-test for dependent samples analysis method is best suited to analyse the results.

The results of the analysis were interpreted by the researcher and are discussed in chapter 4.

## Appendix O - Server Specification

The specification of the server that was used for the experiment is summarised in table 1 below.

**Table 1 Server Specification**

| Field Name       | Data Type                                                |
|------------------|----------------------------------------------------------|
| Operating System | Windows Server 2008 R2 Enterprise Edition Service Pack 1 |
| Processor        | Intel Core i5-2500 3.3GHz (4 core)                       |
| Memory           | 4GB                                                      |
| System type      | 64-bit                                                   |
| Disks            | 1TB (C:)<br>500GB (D:)                                   |

## **Appendix P - SQL Server Configuration**

The version of SQL Server that was installed was SQL Server 2008 R2 SP1 Enterprise Edition. The default configuration for SQL Server was kept with the exception of the data and log files for user databases. These were moved from the C: drive to the D: drive in order to separate the data and log files from the TempDB files.

# Appendix Q - SQL Server T-SQL Scripts

## 1.1 Database Create Scripts

### 1.1.1 TPCE Create Script

```
USE [master]
GO
```

```
/****** Object: Database [TPCE] Script Date: 11/14/2011 22:30:37 *****/
CREATE DATABASE [TPCE] ON PRIMARY
( NAME = N'TPCE', FILENAME = N'D:\SQLDataFiles\MSSQL10.MSSQLSERVER\MSSQL\DATA\TPCE.mdf' , SIZE = 6354368KB , MAXSIZE = UNLIMITED,
FILEGROWTH = 1048576KB )
LOG ON
( NAME = N'TPCE_log', FILENAME = N'D:\SQLDataFiles\MSSQL10.MSSQLSERVER\MSSQL\DATA\TPCE_log.ldf' , SIZE = 13735360KB , MAXSIZE =
2048GB , FILEGROWTH = 102400KB )
GO
```

```
ALTER DATABASE [TPCE] SET COMPATIBILITY_LEVEL = 100
GO
```

```
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
BEGIN
    EXEC [TPCE].[dbo].[sp_fulltext_database] @action = 'enable'
END
GO
```

```
ALTER DATABASE [TPCE] SET ANSI_NULL_DEFAULT OFF
GO
```

```
ALTER DATABASE [TPCE] SET ANSI_NULLS OFF
GO
```

```
ALTER DATABASE [TPCE] SET ANSI_PADDING OFF
```



GO

ALTER DATABASE [TPCE] SET ANSI\_WARNINGS OFF  
GO

ALTER DATABASE [TPCE] SET ARITHABORT OFF  
GO

ALTER DATABASE [TPCE] SET AUTO\_CLOSE OFF  
GO

ALTER DATABASE [TPCE] SET AUTO\_CREATE\_STATISTICS ON  
GO

ALTER DATABASE [TPCE] SET AUTO\_SHRINK OFF  
GO

ALTER DATABASE [TPCE] SET AUTO\_UPDATE\_STATISTICS ON  
GO

ALTER DATABASE [TPCE] SET CURSOR\_CLOSE\_ON\_COMMIT OFF  
GO

ALTER DATABASE [TPCE] SET CURSOR\_DEFAULT GLOBAL  
GO

ALTER DATABASE [TPCE] SET CONCAT\_NULL\_YIELDS\_NULL OFF  
GO

ALTER DATABASE [TPCE] SET NUMERIC\_ROUNDABORT OFF  
GO

ALTER DATABASE [TPCE] SET QUOTED\_IDENTIFIER OFF  
GO

ALTER DATABASE [TPCE] SET RECURSIVE\_TRIGGERS OFF  
GO

ALTER DATABASE [TPCE] SET DISABLE\_BROKER  
GO

```
ALTER DATABASE [TPCE] SET AUTO_UPDATE_STATISTICS_ASYNC OFF  
GO
```

```
ALTER DATABASE [TPCE] SET DATE_CORRELATION_OPTIMIZATION OFF  
GO
```

```
ALTER DATABASE [TPCE] SET TRUSTWORTHY OFF  
GO
```

```
ALTER DATABASE [TPCE] SET ALLOW_SNAPSHOT_ISOLATION OFF  
GO
```

```
ALTER DATABASE [TPCE] SET PARAMETERIZATION SIMPLE  
GO
```

```
ALTER DATABASE [TPCE] SET READ_COMMITTED_SNAPSHOT OFF  
GO
```

```
ALTER DATABASE [TPCE] SET HONOR_BROKER_PRIORITY OFF  
GO
```

```
ALTER DATABASE [TPCE] SET READ_WRITE  
GO
```

```
ALTER DATABASE [TPCE] SET RECOVERY SIMPLE  
GO
```

```
ALTER DATABASE [TPCE] SET MULTI_USER  
GO
```

```
ALTER DATABASE [TPCE] SET PAGE_VERIFY CHECKSUM  
GO
```

```
ALTER DATABASE [TPCE] SET DB_CHAINING OFF  
GO
```

```
EXEC sys.sp_db_vardecimal_storage_format N'TPCE', N'ON'  
GO
```

```
USE [TPCE]
GO
```

```
/***** Object: Table [dbo].[TAXRATE]    Script Date: 11/14/2011 22:30:37 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[TAXRATE](
    [TX_ID] [varchar](4) NOT NULL,
    [TX_NAME] [varchar](50) NOT NULL,
    [TX_RATE] [decimal](6, 5) NOT NULL,
    CONSTRAINT [PK_TAXRATE] PRIMARY KEY CLUSTERED
(
    [TX_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[STATUS_TYPE]    Script Date: 11/14/2011 22:30:37 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[STATUS_TYPE](
    [ST_ID] [varchar](4) NOT NULL,
```

```
        [ST_NAME] [varchar](10) NOT NULL,  
CONSTRAINT [PK_STATUS_TYPE] PRIMARY KEY CLUSTERED  
(  
        [ST_ID] ASC  
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]  
GO
```

```
SET ANSI_PADDING OFF  
GO
```

```
/***** Object: Table [dbo].[TRADE_TYPE]    Script Date: 11/14/2011 22:30:37 *****/
```

```
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
SET ANSI_PADDING ON  
GO
```

```
CREATE TABLE [dbo].[TRADE_TYPE](  
        [TT_ID] [varchar](3) NOT NULL,  
        [TT_NAME] [varchar](12) NOT NULL,  
        [TT_IS_SELL] [bit] NOT NULL,  
        [TT_IS_MRKT] [bit] NOT NULL,  
CONSTRAINT [PK_TRADE_TYPE] PRIMARY KEY CLUSTERED  
(  
        [TT_ID] ASC  
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]  
GO
```

```
SET ANSI_PADDING OFF  
GO
```

```
/***** Object: Table [dbo].[ZIP_CODE]    Script Date: 11/14/2011 22:30:37 *****/
```

```
SET ANSI_NULLS ON
```

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[ZIP_CODE](
    [ZC_CODE] [varchar](12) NOT NULL,
    [ZC_TOWN] [varchar](80) NOT NULL,
    [ZC_DIV] [varchar](80) NOT NULL,
    CONSTRAINT [PK_ZIP_CODE] PRIMARY KEY CLUSTERED
(
    [ZC_CODE] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[SECTOR] Script Date: 11/14/2011 22:30:37 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[SECTOR](
    [SC_ID] [varchar](2) NOT NULL,
    [SC_NAME] [varchar](30) NOT NULL,
    CONSTRAINT [PK_SECTOR] PRIMARY KEY CLUSTERED
(
    [SC_ID] ASC
)
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/****** Object: Table [dbo].[NEWS_ITEM] Script Date: 11/14/2011 22:30:37 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[NEWS_ITEM](
    [NI_ID] [bigint] NOT NULL,
    [NI_HEADLINE] [varchar](80) NOT NULL,
    [NI_SUMMARY] [varchar](255) NOT NULL,
    [NI_ITEM] [varchar](max) NOT NULL,
    [NI_DTS] [datetime2](0) NOT NULL,
    [NI_SOURCE] [varchar](30) NOT NULL,
    [NI_AUTHOR] [varchar](30) NULL,
    CONSTRAINT [PK_NEWS_ITEM] PRIMARY KEY CLUSTERED
(
```

```
    [NI_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/****** Object: Table [dbo].[CHARGE] Script Date: 11/14/2011 22:30:37 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[CHARGE](
    [CH_TT_ID] [varchar](3) NOT NULL,
    [CH_C_TIER] [tinyint] NOT NULL,
    [CH_CHRG] [decimal](10, 2) NOT NULL,
    CONSTRAINT [PK_CHARGE] PRIMARY KEY CLUSTERED
(
    [CH_TT_ID] ASC,
    [CH_C_TIER] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[INDUSTRY]    Script Date: 11/14/2011 22:30:37 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[INDUSTRY](
    [IN_ID] [varchar](2) NOT NULL,
    [IN_NAME] [varchar](50) NOT NULL,
    [IN_SC_ID] [varchar](2) NOT NULL,
    CONSTRAINT [PK_INDUSTRY] PRIMARY KEY CLUSTERED
(
    [IN_ID] ASC

```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[BROKER] Script Date: 11/14/2011 22:30:37 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[BROKER](
    [B_ID] [bigint] NOT NULL,
    [B_ST_ID] [varchar](4) NOT NULL,
    [B_NAME] [varchar](49) NOT NULL,
    [B_NUM_TRADES] [int] NOT NULL,
    [B_COMM_TOTAL] [decimal](12, 2) NOT NULL,
    CONSTRAINT [PK_BROKER] PRIMARY KEY CLUSTERED
(
```

```
    [B_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[ADDRESS] Script Date: 11/14/2011 22:30:37 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
```



GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[ADDRESS](
    [AD_ID] [bigint] NOT NULL,
    [AD_LINE1] [varchar](80) NULL,
    [AD_LINE2] [varchar](80) NULL,
    [AD_ZC_CODE] [varchar](12) NOT NULL,
    [AD_CTRY] [varchar](80) NULL,
    CONSTRAINT [PK_ADDRESS] PRIMARY KEY CLUSTERED
(
    [AD_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[COMPANY] Script Date: 11/14/2011 22:30:37 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[COMPANY](
    [CO_ID] [bigint] NOT NULL,
    [CO_ST_ID] [varchar](4) NOT NULL,
    [CO_NAME] [varchar](60) NOT NULL,
    [CO_IN_ID] [varchar](2) NOT NULL,
    [CO_SP_RATE] [varchar](4) NOT NULL,
    [CO_CEO] [varchar](46) NOT NULL,
    [CO_AD_ID] [bigint] NOT NULL,
```

```

        [CO_DESC] [varchar](150) NOT NULL,
        [CO_OPEN_DATE] [date] NOT NULL,
        CONSTRAINT [PK_COMPANY] PRIMARY KEY CLUSTERED
    (
        [CO_ID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
    DATA_COMPRESSION = PAGE) ON [PRIMARY]
    ) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[CUSTOMER]    Script Date: 11/14/2011 22:30:38 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[CUSTOMER](
    [C_ID] [bigint] NOT NULL,
    [C_TAX_ID] [varchar](20) NOT NULL,
    [C_ST_ID] [varchar](4) NOT NULL,
    [C_L_NAME] [varchar](25) NOT NULL,
    [C_F_NAME] [varchar](20) NOT NULL,
    [C_M_NAME] [varchar](1) NULL,
    [C_GNDR] [varchar](1) NULL,
    [C_TIER] [tinyint] NOT NULL,
    [C_DOB] [date] NOT NULL,
    [C_AD_ID] [bigint] NOT NULL,
    [C_CTRY_1] [varchar](3) NULL,
    [C_AREA_1] [varchar](3) NULL,
    [C_LOCAL_1] [varchar](10) NULL,
    [C_EXT_1] [varchar](5) NULL,
    [C_CTRY_2] [varchar](3) NULL,
    [C_AREA_2] [varchar](3) NULL,

```

```

        [C_LOCAL_2] [varchar](10) NULL,
        [C_EXT_2] [varchar](5) NULL,
        [C_CTRY_3] [varchar](3) NULL,
        [C_AREA_3] [varchar](3) NULL,
        [C_LOCAL_3] [varchar](10) NULL,
        [C_EXT_3] [varchar](5) NULL,
        [C_EMAIL_1] [varchar](50) NULL,
        [C_EMAIL_2] [varchar](50) NULL,
CONSTRAINT [PK_CUSTOMER] PRIMARY KEY CLUSTERED
(
    [C_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[EXCHANGE]    Script Date: 11/14/2011 22:30:38 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[EXCHANGE](
    [EX_ID] [varchar](6) NOT NULL,
    [EX_NAME] [varchar](100) NOT NULL,
    [EX_NUM_SYMB] [int] NOT NULL,
    [EX_OPEN] [smallint] NOT NULL,
    [EX_CLOSE] [smallint] NOT NULL,
    [EX_DESC] [varchar](150) NOT NULL,
    [EX_AD_ID] [bigint] NOT NULL,
CONSTRAINT [PK_EXCHANGE] PRIMARY KEY CLUSTERED
(
    [EX_ID] ASC

```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]  
GO
```

```
SET ANSI_PADDING OFF  
GO
```

```
/***** Object: Table [dbo].[FINANCIAL] Script Date: 11/14/2011 22:30:38 *****/  
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
CREATE TABLE [dbo].[FINANCIAL](  
    [FI_CO_ID] [bigint] NOT NULL,  
    [FI_YEAR] [smallint] NOT NULL,  
    [FI_QTR] [tinyint] NOT NULL,  
    [FI_QTR_START_DATE] [date] NOT NULL,  
    [FI_REVENUE] [decimal](15, 2) NOT NULL,  
    [FI_NET_EARN] [decimal](15, 2) NOT NULL,  
    [FI_BASIC_EPS] [decimal](10, 2) NOT NULL,  
    [FI_DILUT_EPS] [decimal](10, 2) NOT NULL,  
    [FI_MARGIN] [decimal](10, 2) NOT NULL,  
    [FI_INVENTORY] [decimal](15, 2) NOT NULL,  
    [FI_ASSETS] [decimal](15, 2) NOT NULL,  
    [FI_LIABILITY] [decimal](15, 2) NOT NULL,  
    [FI_OUT_BASIC] [bigint] NOT NULL,  
    [FI_OUT_DILUT] [bigint] NOT NULL,  
    CONSTRAINT [PK_FINANCIAL] PRIMARY KEY CLUSTERED  
(  
    [FI_CO_ID] ASC,  
    [FI_YEAR] ASC,  
    [FI_QTR] ASC  
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]  
GO
```

/\*\*\*\*\* Object: Table [dbo].[NEWS\_XREF] Script Date: 11/14/2011 22:30:38 \*\*\*\*\*/

SET ANSI\_NULLS ON  
GO

SET QUOTED\_IDENTIFIER ON  
GO

```
CREATE TABLE [dbo].[NEWS_XREF](
    [NX_NI_ID] [bigint] NOT NULL,
    [NX_CO_ID] [bigint] NOT NULL,
    CONSTRAINT [PK_NEWS_XREF] PRIMARY KEY CLUSTERED
(
    [NX_NI_ID] ASC,
    [NX_CO_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

/\*\*\*\*\* Object: Table [dbo].[WATCH\_LIST] Script Date: 11/14/2011 22:30:38 \*\*\*\*\*/

SET ANSI\_NULLS ON  
GO

SET QUOTED\_IDENTIFIER ON  
GO

```
CREATE TABLE [dbo].[WATCH_LIST](
    [WL_ID] [bigint] NOT NULL,
    [WL_C_ID] [bigint] NOT NULL,
    CONSTRAINT [PK_WATCH_LIST] PRIMARY KEY CLUSTERED
(
    [WL_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

/\*\*\*\*\* Object: Table [dbo].[SECURITY] Script Date: 11/14/2011 22:30:38 \*\*\*\*\*/

SET ANSI\_NULLS ON  
GO

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[SECURITY](
    [S_SYMB] [varchar](15) NOT NULL,
    [S_ISSUE] [varchar](6) NOT NULL,
    [S_ST_ID] [varchar](4) NOT NULL,
    [S_NAME] [varchar](70) NOT NULL,
    [S_EX_ID] [varchar](6) NOT NULL,
    [S_CO_ID] [bigint] NOT NULL,
    [S_NUM_OUT] [bigint] NOT NULL,
    [S_START_DATE] [date] NOT NULL,
    [S_EXCH_DATE] [date] NOT NULL,
    [S_PE] [decimal](10, 2) NOT NULL,
    [S_52WK_HIGH] [decimal](8, 2) NOT NULL,
    [S_52WK_HIGH_DATE] [date] NOT NULL,
    [S_52WK_LOW] [decimal](8, 2) NOT NULL,
    [S_52WK_LOW_DATE] [date] NOT NULL,
    [S_DIVIDEND] [decimal](10, 2) NOT NULL,
    [S_YIELD] [decimal](5, 2) NOT NULL,
    CONSTRAINT [PK_SECURITY] PRIMARY KEY CLUSTERED
(
    [S_SYMB] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[CUSTOMER_TAXRATE]    Script Date: 11/14/2011 22:30:38 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
```

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[CUSTOMER_TAXRATE](
    [CX_TX_ID] [varchar](4) NOT NULL,
    [CX_C_ID] [bigint] NOT NULL,
    CONSTRAINT [PK_CUSTOMER_TAXRATE] PRIMARY KEY CLUSTERED
(
    [CX_TX_ID] ASC,
    [CX_C_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[CUSTOMER\_ACCOUNT] Script Date: 11/14/2011 22:30:38 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[CUSTOMER_ACCOUNT](
    [CA_ID] [bigint] NOT NULL,
    [CA_B_ID] [bigint] NOT NULL,
    [CA_C_ID] [bigint] NOT NULL,
    [CA_NAME] [varchar](50) NOT NULL,
    [CA_TAX_ST] [tinyint] NOT NULL,
    [CA_BAL] [decimal](12, 2) NOT NULL,
    CONSTRAINT [PK_CUSTOMER_ACCOUNT] PRIMARY KEY CLUSTERED
(
    [CA_ID] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[COMPANY_COMPETITOR] Script Date: 11/14/2011 22:30:38 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[COMPANY_COMPETITOR](
    [CP_CO_ID] [bigint] NOT NULL,
    [CP_COMP_CO_ID] [bigint] NOT NULL,
    [CP_IN_ID] [varchar](2) NOT NULL,
    CONSTRAINT [PK_COMPANY_COMPETITOR] PRIMARY KEY CLUSTERED
(
    [CP_CO_ID] ASC,
    [CP_COMP_CO_ID] ASC,
    [CP_IN_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[COMMISSION_RATE] Script Date: 11/14/2011 22:30:38 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
```



GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[COMMISSION_RATE](
    [CR_C_TIER] [tinyint] NOT NULL,
    [CR_TT_ID] [varchar](3) NOT NULL,
    [CR_EX_ID] [varchar](6) NOT NULL,
    [CR_FROM_QTY] [int] NOT NULL,
    [CR_TO_QTY] [int] NOT NULL,
    [CR_RATE] [decimal](5, 2) NOT NULL,
    CONSTRAINT [PK_COMMISSION_RATE] PRIMARY KEY CLUSTERED
(
    [CR_C_TIER] ASC,
    [CR_TT_ID] ASC,
    [CR_EX_ID] ASC,
    [CR_FROM_QTY] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[ACCOUNT\_PERMISSION] Script Date: 11/14/2011 22:30:38 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[ACCOUNT_PERMISSION](
    [AP_CA_ID] [bigint] NOT NULL,
    [AP_ACL] [varchar](4) NOT NULL,
    [AP_TAX_ID] [varchar](20) NOT NULL,
```

```

        [AP_L_NAME] [varchar](25) NOT NULL,
        [AP_F_NAME] [varchar](20) NOT NULL,
CONSTRAINT [PK_ACCOUNT_PERMISSION] PRIMARY KEY CLUSTERED
(
        [AP_CA_ID] ASC,
        [AP_TAX_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[DAILY_MARKET]    Script Date: 11/14/2011 22:30:38 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[DAILY_MARKET](
        [DM_DATE] [date] NOT NULL,
        [DM_S_SYMB] [varchar](15) NOT NULL,
        [DM_CLOSE] [decimal](8, 2) NOT NULL,
        [DM_HIGH] [decimal](8, 2) NOT NULL,
        [DM_LOW] [decimal](8, 2) NOT NULL,
        [DM_VOL] [bigint] NOT NULL,
CONSTRAINT [PK_DAILY_MARKET] PRIMARY KEY CLUSTERED
(
        [DM_DATE] ASC,
        [DM_S_SYMB] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[HOLDING_SUMMARY]    Script Date: 11/14/2011 22:30:38 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[HOLDING_SUMMARY](
    [HS_CA_ID] [bigint] NOT NULL,
    [HS_S_SYMB] [varchar](15) NOT NULL,
    [HS_QTY] [int] NOT NULL,
    CONSTRAINT [PK_HOLDING_SUMMARY] PRIMARY KEY CLUSTERED
(
    [HS_CA_ID] ASC,
    [HS_S_SYMB] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[LAST_TRADE]    Script Date: 11/14/2011 22:30:38 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[LAST_TRADE](
```

```

        [LT_S_SYMB] [varchar](15) NOT NULL,
        [LT_DTS] [datetime2](0) NOT NULL,
        [LT_PRICE] [decimal](8, 2) NOT NULL,
        [LT_OPEN_PRICE] [decimal](8, 2) NOT NULL,
        [LT_VOL] [bigint] NOT NULL,
CONSTRAINT [PK_LAST_TRADE] PRIMARY KEY CLUSTERED
(
    [LT_S_SYMB] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[TRADE]    Script Date: 11/14/2011 22:30:38 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[TRADE](
    [T_ID] [bigint] NOT NULL,
    [T_DTS] [datetime2](0) NOT NULL,
    [T_ST_ID] [varchar](4) NOT NULL,
    [T_TT_ID] [varchar](3) NOT NULL,
    [T_IS_CASH] [bit] NOT NULL,
    [T_S_SYMB] [varchar](15) NOT NULL,
    [T_QTY] [int] NOT NULL,
    [T_BID_PRICE] [decimal](8, 2) NOT NULL,
    [T_CA_ID] [bigint] NOT NULL,
    [T_EXEC_NAME] [varchar](49) NOT NULL,
    [T_TRADE_PRICE] [decimal](8, 2) NULL,
    [T_CHRG] [decimal](10, 2) NOT NULL,
    [T_COMM] [decimal](10, 2) NOT NULL,

```

```

        [T_TAX] [decimal](10, 2) NOT NULL,
        [T_LIFO] [bit] NOT NULL,
    CONSTRAINT [PK_TRADE] PRIMARY KEY CLUSTERED
    (
        [T_ID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
    DATA_COMPRESSION = PAGE) ON [PRIMARY]
    ) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[WATCH_ITEM]    Script Date: 11/14/2011 22:30:38 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[WATCH_ITEM](
    [WI_WL_ID] [bigint] NOT NULL,
    [WI_S_SYMB] [varchar](15) NOT NULL,
    CONSTRAINT [PK_WATCH_ITEM] PRIMARY KEY CLUSTERED
    (
        [WI_WL_ID] ASC,
        [WI_S_SYMB] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
    DATA_COMPRESSION = PAGE) ON [PRIMARY]
    ) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[TRADE_REQUEST]    Script Date: 11/14/2011 22:30:38 *****/
SET ANSI_NULLS ON

```

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[TRADE_REQUEST](
    [TR_T_ID] [bigint] NOT NULL,
    [TR_TT_ID] [varchar](3) NOT NULL,
    [TR_S_SYMB] [varchar](15) NOT NULL,
    [TR_QTY] [int] NOT NULL,
    [TR_BID_PRICE] [decimal](8, 2) NOT NULL,
    [TR_B_ID] [bigint] NOT NULL,
    CONSTRAINT [PK_TRADE_REQUEST] PRIMARY KEY CLUSTERED
(
    [TR_T_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[TRADE\_HISTORY] Script Date: 11/14/2011 22:30:38 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[TRADE_HISTORY](
    [TH_T_ID] [bigint] NOT NULL,
    [TH_DTS] [datetime2](0) NOT NULL,
    [TH_ST_ID] [varchar](4) NOT NULL,
```

```

CONSTRAINT [PK_TRADE_HISTORY] PRIMARY KEY CLUSTERED
(
    [TH_T_ID] ASC,
    [TH_ST_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[SETTLEMENT]    Script Date: 11/14/2011 22:30:38 *****/

```

```

SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[SETTLEMENT](
    [SE_T_ID] [bigint] NOT NULL,
    [SE_CASH_TYPE] [varchar](40) NOT NULL,
    [SE_CASH_DUE_DATE] [date] NOT NULL,
    [SE_AMT] [decimal](10, 2) NOT NULL,
    CONSTRAINT [PK_SETTLEMENT] PRIMARY KEY CLUSTERED
(
    [SE_T_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[CASH_TRANSACTION]    Script Date: 11/14/2011 22:30:38 *****/

```

```

SET ANSI_NULLS ON

```

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[CASH_TRANSACTION](
    [CT_T_ID] [bigint] NOT NULL,
    [CT_DTS] [datetime2](0) NOT NULL,
    [CT_AMT] [decimal](10, 2) NOT NULL,
    [CT_NAME] [varchar](100) NULL,
    CONSTRAINT [PK_CASH_TRANSACTION] PRIMARY KEY CLUSTERED
(
    [CT_T_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[HOLDING\_HISTORY] Script Date: 11/14/2011 22:30:38 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

```
CREATE TABLE [dbo].[HOLDING_HISTORY](
    [HH_H_T_ID] [bigint] NOT NULL,
    [HH_T_ID] [bigint] NOT NULL,
    [HH_BEFORE_QTY] [int] NOT NULL,
    [HH_AFTER_QTY] [int] NOT NULL,
    CONSTRAINT [PK_HOLDING_HISTORY] PRIMARY KEY CLUSTERED
(
    [HH_H_T_ID] ASC,
    [HH_T_ID] ASC
)
```



```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]  
GO
```

```
/***** Object: Table [dbo].[HOLDING] Script Date: 11/14/2011 22:30:38 *****/
```

```
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
SET ANSI_PADDING ON  
GO
```

```
CREATE TABLE [dbo].[HOLDING](  
    [H_T_ID] [bigint] NOT NULL,  
    [H_CA_ID] [bigint] NOT NULL,  
    [H_S_SYMB] [varchar](15) NOT NULL,  
    [H_DTS] [datetime2](0) NOT NULL,  
    [H_PRICE] [decimal](8, 2) NOT NULL,  
    [H_QTY] [int] NOT NULL,  
    CONSTRAINT [PK_HOLDING] PRIMARY KEY CLUSTERED
```

```
(  
    [H_T_ID] ASC  
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]  
GO
```

```
SET ANSI_PADDING OFF  
GO
```

```
/***** Object: ForeignKey [FK_CHARGE_TRADE_TYPE] Script Date: 11/14/2011 22:30:37 *****/
```

```
ALTER TABLE [dbo].[CHARGE] WITH NOCHECK ADD CONSTRAINT [FK_CHARGE_TRADE_TYPE] FOREIGN KEY([CH_TT_ID])  
REFERENCES [dbo].[TRADE_TYPE] ([TT_ID])  
GO
```

```
ALTER TABLE [dbo].[CHARGE] CHECK CONSTRAINT [FK_CHARGE_TRADE_TYPE]  
GO
```

```
/***** Object: ForeignKey [FK_INDUSTY_SECTOR]    Script Date: 11/14/2011 22:30:37 *****/
ALTER TABLE [dbo].[INDUSTY] WITH NOCHECK ADD CONSTRAINT [FK_INDUSTY_SECTOR] FOREIGN KEY([IN_SC_ID])
REFERENCES [dbo].[SECTOR] ([SC_ID])
GO

ALTER TABLE [dbo].[INDUSTY] CHECK CONSTRAINT [FK_INDUSTY_SECTOR]
GO

/***** Object: ForeignKey [FK_BROKER_STATUS_TYPE]    Script Date: 11/14/2011 22:30:37 *****/
ALTER TABLE [dbo].[BROKER] WITH NOCHECK ADD CONSTRAINT [FK_BROKER_STATUS_TYPE] FOREIGN KEY([B_ST_ID])
REFERENCES [dbo].[STATUS_TYPE] ([ST_ID])
GO

ALTER TABLE [dbo].[BROKER] CHECK CONSTRAINT [FK_BROKER_STATUS_TYPE]
GO

/***** Object: ForeignKey [FK_ADDRESS_ZIP_CODE]    Script Date: 11/14/2011 22:30:37 *****/
ALTER TABLE [dbo].[ADDRESS] WITH NOCHECK ADD CONSTRAINT [FK_ADDRESS_ZIP_CODE] FOREIGN KEY([AD_ZC_CODE])
REFERENCES [dbo].[ZIP_CODE] ([ZC_CODE])
GO

ALTER TABLE [dbo].[ADDRESS] CHECK CONSTRAINT [FK_ADDRESS_ZIP_CODE]
GO

/***** Object: ForeignKey [FK_COMPANY_ADDRESS]    Script Date: 11/14/2011 22:30:37 *****/
ALTER TABLE [dbo].[COMPANY] WITH NOCHECK ADD CONSTRAINT [FK_COMPANY_ADDRESS] FOREIGN KEY([CO_AD_ID])
REFERENCES [dbo].[ADDRESS] ([AD_ID])
GO

ALTER TABLE [dbo].[COMPANY] CHECK CONSTRAINT [FK_COMPANY_ADDRESS]
GO

/***** Object: ForeignKey [FK_COMPANY_INDUSTY]    Script Date: 11/14/2011 22:30:37 *****/
ALTER TABLE [dbo].[COMPANY] WITH NOCHECK ADD CONSTRAINT [FK_COMPANY_INDUSTY] FOREIGN KEY([CO_IN_ID])
REFERENCES [dbo].[INDUSTY] ([IN_ID])
GO

ALTER TABLE [dbo].[COMPANY] CHECK CONSTRAINT [FK_COMPANY_INDUSTY]
GO
```

```
/***** Object: ForeignKey [FK_COMPANY_STATUS_TYPE]    Script Date: 11/14/2011 22:30:37 *****/  
ALTER TABLE [dbo].[COMPANY] WITH NOCHECK ADD CONSTRAINT [FK_COMPANY_STATUS_TYPE] FOREIGN KEY([CO_ST_ID])  
REFERENCES [dbo].[STATUS_TYPE] ([ST_ID])  
GO
```

```
ALTER TABLE [dbo].[COMPANY] CHECK CONSTRAINT [FK_COMPANY_STATUS_TYPE]  
GO
```

```
/***** Object: ForeignKey [FK_CUSTOMER_ADDRESS]      Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[CUSTOMER] WITH NOCHECK ADD CONSTRAINT [FK_CUSTOMER_ADDRESS] FOREIGN KEY([C_AD_ID])  
REFERENCES [dbo].[ADDRESS] ([AD_ID])  
GO
```

```
ALTER TABLE [dbo].[CUSTOMER] CHECK CONSTRAINT [FK_CUSTOMER_ADDRESS]  
GO
```

```
/***** Object: ForeignKey [FK_CUSTOMER_STATUS_TYPE]  Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[CUSTOMER] WITH NOCHECK ADD CONSTRAINT [FK_CUSTOMER_STATUS_TYPE] FOREIGN KEY([C_ST_ID])  
REFERENCES [dbo].[STATUS_TYPE] ([ST_ID])  
GO
```

```
ALTER TABLE [dbo].[CUSTOMER] CHECK CONSTRAINT [FK_CUSTOMER_STATUS_TYPE]  
GO
```

```
/***** Object: ForeignKey [FK_EXCHANGE_ADDRESS]      Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[EXCHANGE] WITH NOCHECK ADD CONSTRAINT [FK_EXCHANGE_ADDRESS] FOREIGN KEY([EX_AD_ID])  
REFERENCES [dbo].[ADDRESS] ([AD_ID])  
GO
```

```
ALTER TABLE [dbo].[EXCHANGE] CHECK CONSTRAINT [FK_EXCHANGE_ADDRESS]  
GO
```

```
/***** Object: ForeignKey [FK_FINANCIAL_COMPANY]     Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[FINANCIAL] WITH NOCHECK ADD CONSTRAINT [FK_FINANCIAL_COMPANY] FOREIGN KEY([FI_CO_ID])  
REFERENCES [dbo].[COMPANY] ([CO_ID])  
GO
```

```
ALTER TABLE [dbo].[FINANCIAL] CHECK CONSTRAINT [FK_FINANCIAL_COMPANY]  
GO
```

```
/***** Object: ForeignKey [FK_NEWS_XREF_COMPANY]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[NEWS_XREF] WITH NOCHECK ADD CONSTRAINT [FK_NEWS_XREF_COMPANY] FOREIGN KEY([NX_CO_ID])  
REFERENCES [dbo].[COMPANY] ([CO_ID])  
GO
```

```
ALTER TABLE [dbo].[NEWS_XREF] CHECK CONSTRAINT [FK_NEWS_XREF_COMPANY]  
GO
```

```
/***** Object: ForeignKey [FK_NEWS_XREF_NEWS_ITEM]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[NEWS_XREF] WITH NOCHECK ADD CONSTRAINT [FK_NEWS_XREF_NEWS_ITEM] FOREIGN KEY([NX_NI_ID])  
REFERENCES [dbo].[NEWS_ITEM] ([NI_ID])  
GO
```

```
ALTER TABLE [dbo].[NEWS_XREF] CHECK CONSTRAINT [FK_NEWS_XREF_NEWS_ITEM]  
GO
```

```
/***** Object: ForeignKey [FK_WATCH_LIST_CUSTOMER]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[WATCH_LIST] WITH NOCHECK ADD CONSTRAINT [FK_WATCH_LIST_CUSTOMER] FOREIGN KEY([WL_C_ID])  
REFERENCES [dbo].[CUSTOMER] ([C_ID])  
GO
```

```
ALTER TABLE [dbo].[WATCH_LIST] CHECK CONSTRAINT [FK_WATCH_LIST_CUSTOMER]  
GO
```

```
/***** Object: ForeignKey [FK_SECURITY_COMPANY]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[SECURITY] WITH NOCHECK ADD CONSTRAINT [FK_SECURITY_COMPANY] FOREIGN KEY([S_CO_ID])  
REFERENCES [dbo].[COMPANY] ([CO_ID])  
GO
```

```
ALTER TABLE [dbo].[SECURITY] CHECK CONSTRAINT [FK_SECURITY_COMPANY]  
GO
```

```
/***** Object: ForeignKey [FK_SECURITY_EXCHANGE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[SECURITY] WITH NOCHECK ADD CONSTRAINT [FK_SECURITY_EXCHANGE] FOREIGN KEY([S_EX_ID])  
REFERENCES [dbo].[EXCHANGE] ([EX_ID])  
GO
```

```
ALTER TABLE [dbo].[SECURITY] CHECK CONSTRAINT [FK_SECURITY_EXCHANGE]  
GO
```

```

/***** Object: ForeignKey [FK_SECURITY_STATUS_TYPE]    Script Date: 11/14/2011 22:30:38 *****/
ALTER TABLE [dbo].[SECURITY] WITH NOCHECK ADD CONSTRAINT [FK_SECURITY_STATUS_TYPE] FOREIGN KEY([S_ST_ID])
REFERENCES [dbo].[STATUS_TYPE] ([ST_ID])
GO

ALTER TABLE [dbo].[SECURITY] CHECK CONSTRAINT [FK_SECURITY_STATUS_TYPE]
GO

/***** Object: ForeignKey [FK_CUSTOMER_TAXRATE_CUSTOMER]    Script Date: 11/14/2011 22:30:38 *****/
ALTER TABLE [dbo].[CUSTOMER_TAXRATE] WITH NOCHECK ADD CONSTRAINT [FK_CUSTOMER_TAXRATE_CUSTOMER] FOREIGN KEY([CX_C_ID])
REFERENCES [dbo].[CUSTOMER] ([C_ID])
GO

ALTER TABLE [dbo].[CUSTOMER_TAXRATE] CHECK CONSTRAINT [FK_CUSTOMER_TAXRATE_CUSTOMER]
GO

/***** Object: ForeignKey [FK_CUSTOMER_TAXRATE_TAXRATE]    Script Date: 11/14/2011 22:30:38 *****/
ALTER TABLE [dbo].[CUSTOMER_TAXRATE] WITH NOCHECK ADD CONSTRAINT [FK_CUSTOMER_TAXRATE_TAXRATE] FOREIGN KEY([CX_TX_ID])
REFERENCES [dbo].[TAXRATE] ([TX_ID])
GO

ALTER TABLE [dbo].[CUSTOMER_TAXRATE] CHECK CONSTRAINT [FK_CUSTOMER_TAXRATE_TAXRATE]
GO

/***** Object: ForeignKey [FK_CUSTOMER_ACCOUNT_BROKER]    Script Date: 11/14/2011 22:30:38 *****/
ALTER TABLE [dbo].[CUSTOMER_ACCOUNT] WITH NOCHECK ADD CONSTRAINT [FK_CUSTOMER_ACCOUNT_BROKER] FOREIGN KEY([CA_B_ID])
REFERENCES [dbo].[BROKER] ([B_ID])
GO

ALTER TABLE [dbo].[CUSTOMER_ACCOUNT] CHECK CONSTRAINT [FK_CUSTOMER_ACCOUNT_BROKER]
GO

/***** Object: ForeignKey [FK_CUSTOMER_ACCOUNT_CUSTOMER]    Script Date: 11/14/2011 22:30:38 *****/
ALTER TABLE [dbo].[CUSTOMER_ACCOUNT] WITH NOCHECK ADD CONSTRAINT [FK_CUSTOMER_ACCOUNT_CUSTOMER] FOREIGN KEY([CA_C_ID])
REFERENCES [dbo].[CUSTOMER] ([C_ID])
GO

ALTER TABLE [dbo].[CUSTOMER_ACCOUNT] CHECK CONSTRAINT [FK_CUSTOMER_ACCOUNT_CUSTOMER]
GO
```

```
/***** Object: ForeignKey [FK_COMPANY_COMPETITOR_COMPANY]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[COMPANY_COMPETITOR] WITH NOCHECK ADD CONSTRAINT [FK_COMPANY_COMPETITOR_COMPANY] FOREIGN KEY([CP_CO_ID])  
REFERENCES [dbo].[COMPANY] ([CO_ID])  
GO
```

```
ALTER TABLE [dbo].[COMPANY_COMPETITOR] CHECK CONSTRAINT [FK_COMPANY_COMPETITOR_COMPANY]  
GO
```

```
/***** Object: ForeignKey [FK_COMPANY_COMPETITOR_COMPANY1]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[COMPANY_COMPETITOR] WITH NOCHECK ADD CONSTRAINT [FK_COMPANY_COMPETITOR_COMPANY1] FOREIGN KEY([CP_COMP_CO_ID])  
REFERENCES [dbo].[COMPANY] ([CO_ID])  
GO
```

```
ALTER TABLE [dbo].[COMPANY_COMPETITOR] CHECK CONSTRAINT [FK_COMPANY_COMPETITOR_COMPANY1]  
GO
```

```
/***** Object: ForeignKey [FK_COMPANY_COMPETITOR_INDUSTRY]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[COMPANY_COMPETITOR] WITH NOCHECK ADD CONSTRAINT [FK_COMPANY_COMPETITOR_INDUSTRY] FOREIGN KEY([CP_IN_ID])  
REFERENCES [dbo].[INDUSTRY] ([IN_ID])  
GO
```

```
ALTER TABLE [dbo].[COMPANY_COMPETITOR] CHECK CONSTRAINT [FK_COMPANY_COMPETITOR_INDUSTRY]  
GO
```

```
/***** Object: ForeignKey [FK_COMMISSION_RATE_EXCHANGE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[COMMISSION_RATE] WITH NOCHECK ADD CONSTRAINT [FK_COMMISSION_RATE_EXCHANGE] FOREIGN KEY([CR_EX_ID])  
REFERENCES [dbo].[EXCHANGE] ([EX_ID])  
GO
```

```
ALTER TABLE [dbo].[COMMISSION_RATE] CHECK CONSTRAINT [FK_COMMISSION_RATE_EXCHANGE]  
GO
```

```
/***** Object: ForeignKey [FK_COMMISSION_RATE_TRADE_TYPE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[COMMISSION_RATE] WITH NOCHECK ADD CONSTRAINT [FK_COMMISSION_RATE_TRADE_TYPE] FOREIGN KEY([CR_TT_ID])  
REFERENCES [dbo].[TRADE_TYPE] ([TT_ID])  
GO
```

```
ALTER TABLE [dbo].[COMMISSION_RATE] CHECK CONSTRAINT [FK_COMMISSION_RATE_TRADE_TYPE]  
GO
```

```
/***** Object: ForeignKey [FK_ACCOUNT_PERMISSION_CUSTOMER_ACCOUNT]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[ACCOUNT_PERMISSION] WITH NOCHECK ADD CONSTRAINT [FK_ACCOUNT_PERMISSION_CUSTOMER_ACCOUNT] FOREIGN KEY([AP_CA_ID])  
REFERENCES [dbo].[CUSTOMER_ACCOUNT] ([CA_ID])  
GO
```

```
ALTER TABLE [dbo].[ACCOUNT_PERMISSION] CHECK CONSTRAINT [FK_ACCOUNT_PERMISSION_CUSTOMER_ACCOUNT]  
GO
```

```
/***** Object: ForeignKey [FK_DAILY_MARKET_SECURITY]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[DAILY_MARKET] WITH NOCHECK ADD CONSTRAINT [FK_DAILY_MARKET_SECURITY] FOREIGN KEY([DM_S_SYMB])  
REFERENCES [dbo].[SECURITY] ([S_SYMB])  
GO
```

```
ALTER TABLE [dbo].[DAILY_MARKET] CHECK CONSTRAINT [FK_DAILY_MARKET_SECURITY]  
GO
```

```
/***** Object: ForeignKey [FK_HOLDING_SUMMARY_CUSTOMER_ACCOUNT]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[HOLDING_SUMMARY] WITH NOCHECK ADD CONSTRAINT [FK_HOLDING_SUMMARY_CUSTOMER_ACCOUNT] FOREIGN KEY([HS_CA_ID])  
REFERENCES [dbo].[CUSTOMER_ACCOUNT] ([CA_ID])  
GO
```

```
ALTER TABLE [dbo].[HOLDING_SUMMARY] CHECK CONSTRAINT [FK_HOLDING_SUMMARY_CUSTOMER_ACCOUNT]  
GO
```

```
/***** Object: ForeignKey [FK_HOLDING_SUMMARY_SECURITY]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[HOLDING_SUMMARY] WITH NOCHECK ADD CONSTRAINT [FK_HOLDING_SUMMARY_SECURITY] FOREIGN KEY([HS_S_SYMB])  
REFERENCES [dbo].[SECURITY] ([S_SYMB])  
GO
```

```
ALTER TABLE [dbo].[HOLDING_SUMMARY] CHECK CONSTRAINT [FK_HOLDING_SUMMARY_SECURITY]  
GO
```

```
/***** Object: ForeignKey [FK_LAST_TRADE_SECURITY]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[LAST_TRADE] WITH NOCHECK ADD CONSTRAINT [FK_LAST_TRADE_SECURITY] FOREIGN KEY([LT_S_SYMB])  
REFERENCES [dbo].[SECURITY] ([S_SYMB])  
GO
```

```
ALTER TABLE [dbo].[LAST_TRADE] CHECK CONSTRAINT [FK_LAST_TRADE_SECURITY]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_CUSTOMER_ACCOUNT]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE] WITH NOCHECK ADD CONSTRAINT [FK_TRADE_CUSTOMER_ACCOUNT] FOREIGN KEY([T_CA_ID])  
REFERENCES [dbo].[CUSTOMER_ACCOUNT] ([CA_ID])  
GO
```

```
ALTER TABLE [dbo].[TRADE] CHECK CONSTRAINT [FK_TRADE_CUSTOMER_ACCOUNT]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_SECURITY]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE] WITH NOCHECK ADD CONSTRAINT [FK_TRADE_SECURITY] FOREIGN KEY([T_S_SYMB])  
REFERENCES [dbo].[SECURITY] ([S_SYMB])  
GO
```

```
ALTER TABLE [dbo].[TRADE] CHECK CONSTRAINT [FK_TRADE_SECURITY]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_STATUS_TYPE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE] WITH NOCHECK ADD CONSTRAINT [FK_TRADE_STATUS_TYPE] FOREIGN KEY([T_ST_ID])  
REFERENCES [dbo].[STATUS_TYPE] ([ST_ID])  
GO
```

```
ALTER TABLE [dbo].[TRADE] CHECK CONSTRAINT [FK_TRADE_STATUS_TYPE]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_TRADE_TYPE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE] WITH NOCHECK ADD CONSTRAINT [FK_TRADE_TRADE_TYPE] FOREIGN KEY([T_TT_ID])  
REFERENCES [dbo].[TRADE_TYPE] ([TT_ID])  
GO
```

```
ALTER TABLE [dbo].[TRADE] CHECK CONSTRAINT [FK_TRADE_TRADE_TYPE]  
GO
```

```
/***** Object: ForeignKey [FK_WATCH_ITEM_SECURITY]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[WATCH_ITEM] WITH NOCHECK ADD CONSTRAINT [FK_WATCH_ITEM_SECURITY] FOREIGN KEY([WI_S_SYMB])  
REFERENCES [dbo].[SECURITY] ([S_SYMB])  
GO
```

```
ALTER TABLE [dbo].[WATCH_ITEM] CHECK CONSTRAINT [FK_WATCH_ITEM_SECURITY]  
GO
```



```
/***** Object: ForeignKey [FK_WATCH_ITEM_WATCH_LIST]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[WATCH_ITEM] WITH NOCHECK ADD CONSTRAINT [FK_WATCH_ITEM_WATCH_LIST] FOREIGN KEY([WI_WL_ID])  
REFERENCES [dbo].[WATCH_LIST] ([WL_ID])  
GO
```

```
ALTER TABLE [dbo].[WATCH_ITEM] CHECK CONSTRAINT [FK_WATCH_ITEM_WATCH_LIST]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_REQUEST_BROKER]     Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE_REQUEST] WITH CHECK ADD CONSTRAINT [FK_TRADE_REQUEST_BROKER] FOREIGN KEY([TR_B_ID])  
REFERENCES [dbo].[BROKER] ([B_ID])  
GO
```

```
ALTER TABLE [dbo].[TRADE_REQUEST] CHECK CONSTRAINT [FK_TRADE_REQUEST_BROKER]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_REQUEST_SECURITY]  Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE_REQUEST] WITH CHECK ADD CONSTRAINT [FK_TRADE_REQUEST_SECURITY] FOREIGN KEY([TR_S_SYMB])  
REFERENCES [dbo].[SECURITY] ([S_SYMB])  
GO
```

```
ALTER TABLE [dbo].[TRADE_REQUEST] CHECK CONSTRAINT [FK_TRADE_REQUEST_SECURITY]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_REQUEST_TRADE]     Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE_REQUEST] WITH CHECK ADD CONSTRAINT [FK_TRADE_REQUEST_TRADE] FOREIGN KEY([TR_T_ID])  
REFERENCES [dbo].[TRADE] ([T_ID])  
GO
```

```
ALTER TABLE [dbo].[TRADE_REQUEST] CHECK CONSTRAINT [FK_TRADE_REQUEST_TRADE]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_REQUEST_TRADE_TYPE] Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE_REQUEST] WITH CHECK ADD CONSTRAINT [FK_TRADE_REQUEST_TRADE_TYPE] FOREIGN KEY([TR_TT_ID])  
REFERENCES [dbo].[TRADE_TYPE] ([TT_ID])  
GO
```

```
ALTER TABLE [dbo].[TRADE_REQUEST] CHECK CONSTRAINT [FK_TRADE_REQUEST_TRADE_TYPE]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_HISTORY_STATUS_TYPE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE_HISTORY] WITH NOCHECK ADD CONSTRAINT [FK_TRADE_HISTORY_STATUS_TYPE] FOREIGN KEY([TH_ST_ID])  
REFERENCES [dbo].[STATUS_TYPE] ([ST_ID])  
GO
```

```
ALTER TABLE [dbo].[TRADE_HISTORY] CHECK CONSTRAINT [FK_TRADE_HISTORY_STATUS_TYPE]  
GO
```

```
/***** Object: ForeignKey [FK_TRADE_HISTORY_TRADE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[TRADE_HISTORY] WITH NOCHECK ADD CONSTRAINT [FK_TRADE_HISTORY_TRADE] FOREIGN KEY([TH_T_ID])  
REFERENCES [dbo].[TRADE] ([T_ID])  
GO
```

```
ALTER TABLE [dbo].[TRADE_HISTORY] CHECK CONSTRAINT [FK_TRADE_HISTORY_TRADE]  
GO
```

```
/***** Object: ForeignKey [FK_SETTLEMENT_TRADE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[SETTLEMENT] WITH NOCHECK ADD CONSTRAINT [FK_SETTLEMENT_TRADE] FOREIGN KEY([SE_T_ID])  
REFERENCES [dbo].[TRADE] ([T_ID])  
GO
```

```
ALTER TABLE [dbo].[SETTLEMENT] CHECK CONSTRAINT [FK_SETTLEMENT_TRADE]  
GO
```

```
/***** Object: ForeignKey [FK_CASH_TRANSACTION_TRADE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[CASH_TRANSACTION] WITH NOCHECK ADD CONSTRAINT [FK_CASH_TRANSACTION_TRADE] FOREIGN KEY([CT_T_ID])  
REFERENCES [dbo].[TRADE] ([T_ID])  
GO
```

```
ALTER TABLE [dbo].[CASH_TRANSACTION] CHECK CONSTRAINT [FK_CASH_TRANSACTION_TRADE]  
GO
```

```
/***** Object: ForeignKey [FK_HOLDING_HISTORY_TRADE]    Script Date: 11/14/2011 22:30:38 *****/  
ALTER TABLE [dbo].[HOLDING_HISTORY] WITH NOCHECK ADD CONSTRAINT [FK_HOLDING_HISTORY_TRADE] FOREIGN KEY([HH_H_T_ID])  
REFERENCES [dbo].[TRADE] ([T_ID])  
GO
```

```
ALTER TABLE [dbo].[HOLDING_HISTORY] CHECK CONSTRAINT [FK_HOLDING_HISTORY_TRADE]  
GO
```

```

/***** Object: ForeignKey [FK_HOLDING_HISTORY_TRADE1]    Script Date: 11/14/2011 22:30:38 *****/
ALTER TABLE [dbo].[HOLDING_HISTORY] WITH NOCHECK ADD CONSTRAINT [FK_HOLDING_HISTORY_TRADE1] FOREIGN KEY([HH_T_ID])
REFERENCES [dbo].[TRADE] ([T_ID])
GO

ALTER TABLE [dbo].[HOLDING_HISTORY] CHECK CONSTRAINT [FK_HOLDING_HISTORY_TRADE1]
GO

/***** Object: ForeignKey [FK_HOLDING_HOLDING_SUMMARY]    Script Date: 11/14/2011 22:30:38 *****/
ALTER TABLE [dbo].[HOLDING] WITH NOCHECK ADD CONSTRAINT [FK_HOLDING_HOLDING_SUMMARY] FOREIGN KEY([H_CA_ID], [H_S_SYMB])
REFERENCES [dbo].[HOLDING_SUMMARY] ([HS_CA_ID], [HS_S_SYMB])
GO

ALTER TABLE [dbo].[HOLDING] CHECK CONSTRAINT [FK_HOLDING_HOLDING_SUMMARY]
GO

/***** Object: ForeignKey [FK_HOLDING_TRADE]    Script Date: 11/14/2011 22:30:38 *****/
ALTER TABLE [dbo].[HOLDING] WITH NOCHECK ADD CONSTRAINT [FK_HOLDING_TRADE] FOREIGN KEY([H_T_ID])
REFERENCES [dbo].[TRADE] ([T_ID])
GO

ALTER TABLE [dbo].[HOLDING] CHECK CONSTRAINT [FK_HOLDING_TRADE]
GO

```

### 1.1.2 TPCE\_DM Create Script

```

USE [master]
GO

/***** Object: Database [TPCE_DM]    Script Date: 11/18/2011 00:40:38 *****/
CREATE DATABASE [TPCE_DM] ON PRIMARY
( NAME = N'TPCE_DM', FILENAME = N'D:\SQLDataFiles\MSSQL10.MSSQLSERVER\MSSQL\DATA\TPCE_DM.mdf' , SIZE = 6291456KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 1048576KB )
LOG ON

```

```
( NAME = N'TPCE_DM_log', FILENAME = N'C:\SQLLogFiles\MSSQL10.MSSQLSERVER\MSSQL\DATA\TPCE_DM_log.ldf' , SIZE = 2097152KB , MAXSIZE = 2048GB , FILEGROWTH = 102400KB )  
GO
```

```
ALTER DATABASE [TPCE_DM] SET COMPATIBILITY_LEVEL = 100  
GO
```

```
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))  
BEGIN  
    EXEC [TPCE_DM].[dbo].[sp_fulltext_database] @action = 'enable'  
END  
GO
```

```
ALTER DATABASE [TPCE_DM] SET ANSI_NULL_DEFAULT OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET ANSI_NULLS OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET ANSI_PADDING OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET ANSI_WARNINGS OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET ARITHABORT OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET AUTO_CLOSE OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET AUTO_CREATE_STATISTICS ON  
GO
```

```
ALTER DATABASE [TPCE_DM] SET AUTO_SHRINK OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET AUTO_UPDATE_STATISTICS ON  
GO
```

```
ALTER DATABASE [TPCE_DM] SET CURSOR_CLOSE_ON_COMMIT OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET CURSOR_DEFAULT GLOBAL  
GO
```

```
ALTER DATABASE [TPCE_DM] SET CONCAT_NULL_YIELDS_NULL OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET NUMERIC_ROUNDABORT OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET QUOTED_IDENTIFIER OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET RECURSIVE_TRIGGERS OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET DISABLE_BROKER  
GO
```

```
ALTER DATABASE [TPCE_DM] SET AUTO_UPDATE_STATISTICS_ASYNC OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET DATE_CORRELATION_OPTIMIZATION OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET TRUSTWORTHY OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET ALLOW_SNAPSHOT_ISOLATION OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET PARAMETERIZATION SIMPLE  
GO
```

```
ALTER DATABASE [TPCE_DM] SET READ_COMMITTED_SNAPSHOT OFF  
GO
```

```
ALTER DATABASE [TPCE_DM] SET HONOR_BROKER_PRIORITY OFF
```

GO

ALTER DATABASE [TPCE\_DM] SET RECOVERY SIMPLE  
GO

ALTER DATABASE [TPCE\_DM] SET MULTI\_USER  
GO

ALTER DATABASE [TPCE\_DM] SET PAGE\_VERIFY CHECKSUM  
GO

ALTER DATABASE [TPCE\_DM] SET DB\_CHAINING OFF  
GO

EXEC sys.sp\_db\_vardecimal\_storage\_format N'TPCE\_DM', N'ON'  
GO

USE [TPCE\_DM]  
GO

/\*\*\*\*\* Object: Table [dbo].[D\_Broker] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/  
SET ANSI\_NULLS ON  
GO

SET QUOTED\_IDENTIFIER ON  
GO

SET ANSI\_PADDING ON  
GO

CREATE TABLE [dbo].[D\_Broker](  
 [Broker\_SK] [int] IDENTITY(1,1) NOT NULL,  
 [BrokerID] [bigint] NOT NULL,  
 [BrokerName] [varchar](49) NOT NULL,  
 [BrokerStatusCode] [varchar](4) NOT NULL,  
 [BrokerStatusDesc] [varchar](10) NOT NULL,  
 [WH\_DateInserted] [datetime2](0) NOT NULL,  
 [WH\_DateUpdated] [datetime2](0) NULL,  
 CONSTRAINT [PK\_D\_Broker] PRIMARY KEY CLUSTERED  
(

```
        [Broker_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
```

GO

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[D_CashType]    Script Date: 2013-02-13 06:14:00 AM *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[D_CashType](
    [CashType_SK] [int] IDENTITY(1,1) NOT NULL,
    [CashType] [varchar](12) NOT NULL,
    [IsCashFlag] [bit] NOT NULL,
    [WH_DateInserted] [datetime2](0) NOT NULL,
    [WH_DateUpdated] [datetime2](0) NULL,
    CONSTRAINT [PK_D_CashType] PRIMARY KEY CLUSTERED
(
    [CashType_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
```

GO

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[D_Company]    Script Date: 2013-02-13 06:14:00 AM *****/
SET ANSI_NULLS ON
```

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[D_Company](
    [Company_SK] [int] IDENTITY(1,1) NOT NULL,
    [CompanyID] [bigint] NOT NULL,
    [CompanyName] [varchar](60) NOT NULL,
    [CompanyDesc] [varchar](150) NOT NULL,
    [CompanyStatusCode] [varchar](4) NOT NULL,
    [CompanyStatusDesc] [varchar](10) NOT NULL,
    [CompanyCEO] [varchar](46) NOT NULL,
    [CompanyCreditRating] [varchar](4) NOT NULL,
    [CompanyOpenDate] [date] NOT NULL,
    [IndustryCode] [varchar](2) NOT NULL,
    [IndustryDesc] [varchar](50) NOT NULL,
    [SectorCode] [varchar](2) NOT NULL,
    [SectorDesc] [varchar](30) NOT NULL,
    [AddressLine1] [varchar](80) NOT NULL,
    [AddressLine2] [varchar](80) NULL,
    [TownName] [varchar](80) NOT NULL,
    [StateProvinceCode] [varchar](80) NOT NULL,
    [PostalCode] [varchar](12) NOT NULL,
    [CountryName] [varchar](80) NOT NULL,
    [WH_StartDate] [datetime2](0) NOT NULL,
    [WH_EndDate] [datetime2](0) NULL,
    [WH_CurrInd] [bit] NOT NULL,
    CONSTRAINT [PK_D_Company] PRIMARY KEY CLUSTERED
(
    [Company_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
```

GO



```
SET ANSI_PADDING OFF
GO
```

```
/****** Object: Table [dbo].[D_CustomerAccount]    Script Date: 2013-02-13 06:14:00 AM *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[D_CustomerAccount](
    [CustomerAccount_SK] [int] IDENTITY(1,1) NOT NULL,
    [AccountID] [bigint] NOT NULL,
    [AccountName] [varchar](50) NOT NULL,
    [TaxStatusCode] [tinyint] NOT NULL,
    [TaxStatusDesc] [varchar](30) NOT NULL,
    [CustomerID] [bigint] NOT NULL,
    [LastName] [varchar](25) NOT NULL,
    [FirstName] [varchar](20) NOT NULL,
    [MiddleInitial] [char](1) NOT NULL,
    [GENDER] [varchar](7) NOT NULL,
    [DateOfBirth] [date] NOT NULL,
    [TierNumber] [tinyint] NOT NULL,
    [TierDesc] [varchar](35) NOT NULL,
    [PhonePrimary] [varchar](30) NOT NULL,
    [PhoneSecondary] [varchar](30) NOT NULL,
    [EmailPrimary] [varchar](50) NOT NULL,
    [EmailSecondary] [varchar](50) NOT NULL,
    [AddressLine1] [varchar](80) NOT NULL,
    [AddressLine2] [varchar](80) NOT NULL,
    [TownName] [varchar](80) NOT NULL,
    [StateProvinceCode] [varchar](80) NOT NULL,
    [PostalCode] [varchar](12) NOT NULL,
    [CountryName] [varchar](80) NOT NULL,
    [CustomerStatusCode] [varchar](4) NOT NULL,
    [CustomerStatusDesc] [varchar](10) NOT NULL,
    [WH_StartDate] [datetime2](0) NOT NULL,
```

```

        [WH_EndDate] [datetime2](0) NULL,
        [WH_CurrInd] [bit] NOT NULL,
    CONSTRAINT [PK_D_CustomerAccount] PRIMARY KEY CLUSTERED
    (
        [CustomerAccount_SK] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
    DATA_COMPRESSION = PAGE) ON [PRIMARY]
    ) ON [PRIMARY]

GO

SET ANSI_PADDING OFF
GO

/***** Object: Table [dbo].[D_Date]    Script Date: 2013-02-13 06:14:00 AM *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[D_Date](
    [Date_SK] [int] NOT NULL,
    [Date] [date] NOT NULL,
    [DayOfWeekNumber] [tinyint] NOT NULL,
    [DayOfWeekName] [varchar](10) NOT NULL,
    [DayOfMonthNumber] [tinyint] NOT NULL,
    [DayOfYearNumber] [smallint] NOT NULL,
    [MonthName] [varchar](10) NOT NULL,
    [MonthOfYearNumber] [tinyint] NOT NULL,
    [Quarter] [tinyint] NOT NULL,
    [Year] [smallint] NOT NULL,
    [WH_DateInserted] [datetime2](0) NOT NULL,
    [WH_DateUpdated] [datetime2](0) NULL,
    CONSTRAINT [PK_D_Date] PRIMARY KEY CLUSTERED
    (
        [Date_SK] ASC

```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]
```

GO

```
SET ANSI_PADDING OFF  
GO
```

```
/***** Object: Table [dbo].[D_Exchange]    Script Date: 2013-02-13 06:14:00 AM *****/  
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
SET ANSI_PADDING ON  
GO
```

```
CREATE TABLE [dbo].[D_Exchange](  
    [Exchange_SK] [int] IDENTITY(1,1) NOT NULL,  
    [ExchangeID] [varchar](6) NOT NULL,  
    [ExchangeName] [varchar](100) NOT NULL,  
    [ExchangeDesc] [varchar](150) NOT NULL,  
    [OpeningTime] [time](0) NOT NULL,  
    [ClosingTime] [time](0) NOT NULL,  
    [ListedSecurities] [int] NOT NULL,  
    [AddressLine1] [varchar](80) NOT NULL,  
    [AddressLine2] [varchar](80) NULL,  
    [TownName] [varchar](80) NOT NULL,  
    [StateProvinceCode] [varchar](80) NOT NULL,  
    [PostalCode] [varchar](12) NOT NULL,  
    [CountryName] [varchar](80) NOT NULL,  
    [WH_DateInserted] [datetime2](0) NOT NULL,  
    [WH_DateUpdated] [datetime2](0) NULL,  
    CONSTRAINT [PK_D_Exchange] PRIMARY KEY CLUSTERED  
    (  
        [Exchange_SK] ASC  
    )  
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]
```

```
) ON [PRIMARY]
```

```
GO
```

```
SET ANSI_PADDING OFF
```

```
GO
```

```
/****** Object: Table [dbo].[D_FiscalQuarter]    Script Date: 2013-02-13 06:14:00 AM *****/
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
CREATE TABLE [dbo].[D_FiscalQuarter](
    [FiscalQuarter_SK] [int] NOT NULL,
    [FiscalQuarter] [tinyint] NOT NULL,
    [FiscalYear] [smallint] NOT NULL,
    [WH_DateInserted] [datetime2](0) NOT NULL,
    [WH_DateUpdated] [datetime2](0) NULL,
    CONSTRAINT [PK_D_Quarter] PRIMARY KEY CLUSTERED
(
    [FiscalQuarter_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO
```

```
/****** Object: Table [dbo].[D_Security]    Script Date: 2013-02-13 06:14:00 AM *****/
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
SET ANSI_PADDING ON
```

```
GO
```

```
CREATE TABLE [dbo].[D_Security](
```

```

[Security_SK] [int] IDENTITY(1,1) NOT NULL,
[SecurityCode] [varchar](15) NOT NULL,
[SecurityName] [varchar](70) NOT NULL,
[IssueType] [varchar](6) NOT NULL,
[FirstTradeDate] [date] NOT NULL,
[SecurityStatusCode] [varchar](4) NOT NULL,
[SecurityStatusDesc] [varchar](10) NOT NULL,
[WH_DateInserted] [datetime2](0) NOT NULL,
[WH_DateUpdated] [datetime2](0) NULL,
CONSTRAINT [PK_D_Security] PRIMARY KEY CLUSTERED
(
    [Security_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]

```

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[D\_TradeStatus] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```

CREATE TABLE [dbo].[D_TradeStatus](
    [TradeStatus_SK] [int] IDENTITY(1,1) NOT NULL,
    [TradeStatusCode] [varchar](4) NOT NULL,
    [TradeStatusDesc] [varchar](10) NOT NULL,
    [WH_DateInserted] [datetime2](0) NOT NULL,
    [WH_DateUpdated] [datetime2](0) NULL,
    CONSTRAINT [PK_D_TradeStatus] PRIMARY KEY CLUSTERED
(
    [TradeStatus_SK] ASC
)

```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]
```

GO

```
SET ANSI_PADDING OFF  
GO
```

```
/***** Object: Table [dbo].[D_TradeType]    Script Date: 2013-02-13 06:14:00 AM *****/  
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
SET ANSI_PADDING ON  
GO
```

```
CREATE TABLE [dbo].[D_TradeType](  
    [TradeType_SK] [int] IDENTITY(1,1) NOT NULL,  
    [TradeTypeCode] [varchar](3) NOT NULL,  
    [TradeTypeDesc] [varchar](12) NOT NULL,  
    [TradeCateGO  
ry] [varchar](4) NOT NULL,  
    [MarketTransactionFlag] [bit] NOT NULL,  
    [WH_DateInserted] [datetime2](0) NOT NULL,  
    [WH_DateUpdated] [datetime2](0) NULL,  
    CONSTRAINT [PK_D_TradeType] PRIMARY KEY CLUSTERED
```

```
(  
    [TradeType_SK] ASC  
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]
```

GO

```
SET ANSI_PADDING OFF  
GO
```

/\*\*\*\*\* Object: Table [dbo].[F\_DailyMarketPerformance] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

SET ANSI\_NULLS ON  
GO

SET QUOTED\_IDENTIFIER ON  
GO

```
CREATE TABLE [dbo].[F_DailyMarketPerformance](
    [Date_SK] [int] NOT NULL,
    [Company_SK] [int] NOT NULL,
    [Exchange_SK] [int] NOT NULL,
    [Security_SK] [int] NOT NULL,
    [DayClose] [money] NOT NULL,
    [DayHigh] [money] NOT NULL,
    [DayLow] [money] NOT NULL,
    [DayVolume] [bigint] NOT NULL,
    [2WeekHigh] [money] NULL,
    [2WeekLow] [money] NULL,
    [2WeekVolume] [bigint] NULL,
    [WH_DateInserted] [datetime2](0) NOT NULL
) ON [PRIMARY]
WITH
(
    DATA_COMPRESSION = PAGE
)
```

GO

/\*\*\*\*\* Object: Table [dbo].[F\_QuarterlyCompanyPerformance] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

SET ANSI\_NULLS ON  
GO

SET QUOTED\_IDENTIFIER ON  
GO

```
CREATE TABLE [dbo].[F_QuarterlyCompanyPerformance](
    [FiscalQuarter_SK] [int] NOT NULL,
    [Company_SK] [int] NOT NULL,
    [FiscalQuarterStartDate_SK] [int] NOT NULL,
    [Revenue] [money] NOT NULL,
```

```

[NetEarnings] [money] NOT NULL,
[EarningsPerShareBasic] [money] NOT NULL,
[EarningsPerShareDiluted] [money] NOT NULL,
[Margin] [money] NOT NULL,
[Inventory] [money] NOT NULL,
[Assets] [money] NOT NULL,
[Liabilities] [money] NOT NULL,
[SharesOutstandingBasicAvg] [bigint] NOT NULL,
[SharesOutstandingDilutedAvg] [bigint] NOT NULL,
[WH_DateInserted] [datetime2](0) NOT NULL
) ON [PRIMARY]
WITH
(
DATA_COMPRESSION = PAGE
)

```

GO

/\*\*\*\*\* Object: Table [dbo].[F\_Trade] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

```

SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

CREATE TABLE [dbo].[F_Trade](
    [Date_SK] [int] NOT NULL,
    [Broker_SK] [int] NOT NULL,
    [CustomerAccount_SK] [int] NOT NULL,
    [Company_SK] [int] NOT NULL,
    [Exchange_SK] [int] NOT NULL,
    [Security_SK] [int] NOT NULL,
    [TradeType_SK] [int] NOT NULL,
    [CashType_SK] [int] NOT NULL,
    [TradeStatus_SK] [int] NOT NULL,
    [TradeID] [bigint] NOT NULL,
    [TradeDateTime] [datetime2](0) NOT NULL,
    [Quantity] [int] NOT NULL,
    [BidPrice] [money] NOT NULL,
    [TradePrice] [money] NOT NULL,

```



```
        [Charge] [money] NOT NULL,  
        [Commission] [money] NOT NULL,  
        [Tax] [money] NOT NULL,  
        [WH_DateInserted] [datetime2](0) NOT NULL  
    ) ON [PRIMARY]  
WITH  
(  
    DATA_COMPRESSION = PAGE  
)
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_D\_Broker] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_D\_Broker]

AS

SELECT

```
    b.B_ID AS BrokerID,  
    b.B_NAME AS BrokerName,  
    s.ST_ID AS BrokerStatusCode,  
    s.ST_NAME AS BrokerStatusDesc
```

FROM

```
    TPCE.dbo.[BROKER] b  
    JOIN TPCE.dbo.STATUS_TYPE s  
        ON s.ST_ID = b.B_ST_ID;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_D\_CashType] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

```

CREATE VIEW [dbo].[V_D_CashType]
AS
SELECT DISTINCT
    CASE t.T_IS_CASH
        WHEN 1 THEN 'Cash Account'
        WHEN 0 THEN 'Margin'
    END AS CashType,
    t.T_IS_CASH AS IsCashFlag
FROM
    TPCE.dbo.TRADE t;

```

GO

/\*\*\*\*\* Object: View [dbo].[V\_D\_Company] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

```

SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

CREATE VIEW [dbo].[V_D_Company]
AS
SELECT
    c.CO_ID AS CompanyID,
    c.CO_NAME AS CompanyName,
    c.CO_DESC AS CompanyDesc,
    st.ST_ID AS CompanyStatusCode,
    st.ST_NAME AS CompanyStatusDesc,
    c.CO_CEO AS CompanyCEO,
    c.CO_SP_RATE AS CompanyCreditRating,
    c.CO_OPEN_DATE AS CompanyOpenDate,
    i.IN_ID AS IndustryCode,
    i.IN_NAME AS IndustryDesc,
    s.SC_ID AS SectorCode,
    s.SC_NAME AS SectorDesc,
    a.AD_LINE1 AS AddressLine1,
    ISNULL(a.AD_LINE2, '') AS AddressLine2,
    z.ZC_TOWN AS TownName,
    z.ZC_DIV AS StateProvinceCode,
    z.ZC_CODE AS PostalCode,

```

```

CASE UPPER(a.AD_CTRY)
    WHEN 'USA' THEN UPPER(a.AD_CTRY)
    WHEN NULL THEN 'Unknown'
ELSE
    UPPER(LEFT(a.AD_CTRY,1)) + LOWER(RIGHT(a.AD_CTRY,LEN(a.AD_CTRY)-1))
END AS CountryName
FROM
    TPCE.dbo.COMPANY c
    JOIN TPCE.dbo.INDUSTRY i
        ON c.CO_IN_ID = i.IN_ID
    JOIN TPCE.dbo.SECTOR s
        ON i.IN_SC_ID = s.SC_ID
    JOIN TPCE.dbo.STATUS_TYPE st
        ON c.CO_ST_ID = st.ST_ID
    JOIN TPCE.dbo.[ADDRESS] a
        ON c.CO_AD_ID = a.AD_ID
    JOIN TPCE.dbo.ZIP_CODE z
        ON a.AD_ZC_CODE = z.ZC_CODE;

GO

/***** Object: View [dbo].[V_D_CustomerAccount]    Script Date: 2013-02-13 06:14:00 AM *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[V_D_CustomerAccount]
AS
SELECT
    a.CA_ID AS AccountID,
    a.CA_NAME AS AccountName,
    a.CA_TAX_ST AS TaxStatusCode,
    CASE a.CA_TAX_ST
        WHEN 0 THEN 'Not taxable.'
        WHEN 1 THEN 'Taxable. Withhold tax.'
        WHEN 2 THEN 'Taxable. Do not withhold tax.'
    ELSE
        'Unknown'

```

```

END AS TaxStatusDesc,
c.C_ID AS CustomerID,
c.C_L_NAME AS LastName,
c.C_F_NAME AS FirstName,
ISNULL(c.C_M_NAME, '') AS MiddleInitial,
CASE c.C_GNDR
    WHEN 'M' THEN 'Male'
    WHEN 'F' THEN 'Female'
    ELSE 'Unknown'
END AS GENDER,
c.C_DOB AS DateOfBirth,
c.C_TIER AS TierNumber,
CASE c.C_TIER
    WHEN 1 THEN 'Has between 1 and 4 accounts.'
    WHEN 2 THEN 'Has between 2 and 8 accounts.'
    WHEN 3 THEN 'Has between 5 and 10 accounts.'
    ELSE 'Unknown'
END AS TierDesc,
CASE
    WHEN c.C_CTRY_1 IS NULL AND c.C_AREA_1 IS NULL AND c.C_LOCAL_1 IS NULL
        THEN 'Unknown'
    WHEN c.C_EXT_1 IS NULL
        THEN c.C_CTRY_1 + '-' + c.C_AREA_1 + '-' + LEFT(c.C_LOCAL_1,3) + '-' + RIGHT(c.C_LOCAL_1,4)
    ELSE
        c.C_CTRY_1 + '-' + c.C_AREA_1 + '-' + LEFT(c.C_LOCAL_1,3) + '-' + RIGHT(c.C_LOCAL_1,4) + ' ext. ' + c.C_EXT_1
END AS PhonePrimary,
CASE
    WHEN c.C_CTRY_2 IS NULL AND c.C_AREA_2 IS NULL AND c.C_LOCAL_2 IS NULL
        THEN 'Unknown'
    WHEN c.C_EXT_2 IS NULL
        THEN c.C_CTRY_2 + '-' + c.C_AREA_2 + '-' + LEFT(c.C_LOCAL_2,3) + '-' + RIGHT(c.C_LOCAL_2,4)
    ELSE
        c.C_CTRY_2 + '-' + c.C_AREA_2 + '-' + LEFT(c.C_LOCAL_2,3) + '-' + RIGHT(c.C_LOCAL_2,4) + ' ext. ' + c.C_EXT_2
END AS PhoneSecondary,
ISNULL(c.C_EMAIL_1, 'Unknown') AS EmailPrimary,
ISNULL(c.C_EMAIL_2, 'Unknown') AS EmailSecondary,
ad.AD_LINE1 AS AddressLine1,
ISNULL(ad.AD_LINE2, '') AS AddressLine2,
z.ZC_TOWN AS TownName,
z.ZC_DIV AS StateProvinceCode,

```

```

        ad.AD_ZC_CODE AS PostalCode,
        CASE UPPER(ad.AD_CTRY)
            WHEN 'USA' THEN UPPER(ad.AD_CTRY)
            WHEN NULL THEN 'Unknown'
        ELSE
            UPPER(LEFT(ad.AD_CTRY,1)) + LOWER(RIGHT(ad.AD_CTRY,LEN(ad.AD_CTRY)-1))
        END AS CountryName,
        st.ST_ID AS CustomerStatusCode,
        st.ST_NAME AS CustomerStatusDesc
FROM
    TPCE.dbo.CUSTOMER_ACCOUNT a
    JOIN TPCE.dbo.CUSTOMER c
        ON c.C_ID = a.CA_C_ID
    JOIN TPCE.dbo.[ADDRESS] ad
        ON c.C_AD_ID = ad.AD_ID
    JOIN TPCE.dbo.ZIP_CODE z
        ON ad.AD_ZC_CODE = z.ZC_CODE
    JOIN TPCE.dbo.STATUS_TYPE st
        ON c.C_ST_ID = st.ST_ID;

GO

/***** Object: View [dbo].[V_D_Exchange]    Script Date: 2013-02-13 06:14:00 AM *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[V_D_Exchange]
AS
SELECT
    e.EX_ID AS ExchangeID,
    e.EX_NAME AS ExchangeName,
    e.EX_DESC AS ExchangeDesc,
    CASE LEN(e.EX_OPEN)
        WHEN 3 THEN '0' + LEFT(CONVERT(CHAR(3),e.EX_OPEN),1) + ':' + RIGHT(CONVERT(CHAR(3),e.EX_OPEN),2)
        WHEN 4 THEN LEFT(CONVERT(CHAR(4),e.EX_OPEN),2) + ':' + RIGHT(CONVERT(CHAR(4),e.EX_OPEN),2)
    END AS OpeningTime,
    CASE LEN(e.EX_CLOSE)

```

```

        WHEN 3 THEN '0' + LEFT(CONVERT(CHAR(3),e.EX_CLOSE),1) + ':' + RIGHT(CONVERT(CHAR(3),e.EX_CLOSE),2)
        WHEN 4 THEN LEFT(CONVERT(CHAR(4),e.EX_CLOSE),2) + ':' + RIGHT(CONVERT(CHAR(4),e.EX_CLOSE),2)
    END AS ClosingTime,
    e.EX_NUM_SYMB AS ListedSecurities,
    a.AD_LINE1 AS AddressLine1,
    ISNULL(a.AD_LINE2, '') AS AddressLine2,
    z.ZC_TOWN AS TownName,
    z.ZC_DIV AS StateProvinceCode,
    z.ZC_CODE AS PostalCode,
    CASE UPPER(a.AD_CTRY)
        WHEN 'USA' THEN UPPER(a.AD_CTRY)
        WHEN NULL THEN 'Unknown'
    ELSE
        UPPER(LEFT(a.AD_CTRY,1)) + LOWER(RIGHT(a.AD_CTRY,LEN(a.AD_CTRY)-1))
    END AS CountryName
FROM
    TPCE.dbo.EXCHANGE e
JOIN TPCE.dbo.[ADDRESS] a
    ON e.EX_AD_ID = a.AD_ID
JOIN TPCE.dbo.ZIP_CODE z
    ON a.AD_ZC_CODE = z.ZC_CODE;

GO

/***** Object: View [dbo].[V_D_Security]    Script Date: 2013-02-13 06:14:00 AM *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[V_D_Security]
AS
SELECT
    s.S_SYMB AS SecurityCode,
    s.S_NAME AS SecurityName,
    s.S_ISSUE AS IssueType,
    s.S_START_DATE AS FirstTradeDate,
    st.ST_ID AS SecurityStatusCode,
    st.ST_NAME AS SecurityStatusDesc

```

```
FROM
    TPCE.dbo.[SECURITY] s
    JOIN TPCE.dbo.STATUS_TYPE st
        ON s.S_ST_ID = st.ST_ID;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_D\_TradeStatus] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_D\_TradeStatus]

AS

SELECT

st.ST\_ID AS TradeStatusCode,  
 st.ST\_NAME AS TradeStatusDesc

FROM

TPCE.dbo.STATUS\_TYPE st;

GO

/\*\*\*\*\* Object: View [dbo].[V\_D\_TradeType] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_D\_TradeType]

AS

SELECT

tt.TT\_ID AS TradeTypeCode,  
 tt.TT\_NAME AS TradeTypeDesc,  
 CASE tt.TT\_IS\_SELL  
 WHEN 0 THEN 'Buy'  
 ELSE 'Sell'

END

AS

TradeCateGO

ry,

tt.TT\_IS\_MRKT AS MarketTransactionFlag

FROM

TPCE.dbo.TRADE\_TYPE tt;

GO

/\*\*\*\*\*\* Object: Index [IDX\_F\_DailyMarketPerformance\_Date\_SK] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE CLUSTERED INDEX [IDX\_F\_DailyMarketPerformance\_Date\_SK] ON [dbo].[F\_DailyMarketPerformance]

(  
[Date\_SK] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]  
GO

/\*\*\*\*\*\* Object: Index [IDX\_F\_QuarterlyCompanyPerformance\_FiscalQuarter\_SK] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE CLUSTERED INDEX [IDX\_F\_QuarterlyCompanyPerformance\_FiscalQuarter\_SK] ON [dbo].[F\_QuarterlyCompanyPerformance]

(  
[FiscalQuarter\_SK] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]  
GO

/\*\*\*\*\*\* Object: Index [IDX\_F\_Trade\_Date\_SK] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE CLUSTERED INDEX [IDX\_F\_Trade\_Date\_SK] ON [dbo].[F\_Trade]

(  
[Date\_SK] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]  
GO

/\*\*\*\*\*\* Object: Index [IXD\_D\_Broker\_BrokerID] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IXD\_D\_Broker\_BrokerID] ON [dbo].[D\_Broker]

(  
[BrokerID] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]  
GO



```

/***** Object: Index [IDX_D_CashType_IsCashFlag]    Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_D_CashType_IsCashFlag] ON [dbo].[D_CashType]
(
    [IsCashFlag] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Index [IDX_D_Company_CompanyID]      Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_D_Company_CompanyID] ON [dbo].[D_Company]
(
    [CompanyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Index [IDX_D_Account_AccountID]      Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_D_Account_AccountID] ON [dbo].[D_CustomerAccount]
(
    [AccountID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Index [IDX_D_Date_Date]             Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_D_Date_Date] ON [dbo].[D_Date]
(
    [Date] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

SET ANSI_PADDING ON

GO

/***** Object: Index [IDX_D_Exchange_ExchangeID]   Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_D_Exchange_ExchangeID] ON [dbo].[D_Exchange]
(
    [ExchangeID] ASC

```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/****** Object: Index [IDX_D_FiscalQuarter_FiscalQuarter_FiscalYear] Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_D_FiscalQuarter_FiscalQuarter_FiscalYear] ON [dbo].[D_FiscalQuarter]
```

```
(
    [FiscalQuarter] ASC,
    [FiscalYear] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING ON
```

```
GO
```

```
/****** Object: Index [IDX_D_Security_SecurityCode] Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_D_Security_SecurityCode] ON [dbo].[D_Security]
```

```
(
    [SecurityCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING ON
```

```
GO
```

```
/****** Object: Index [IDX_D_TradeStatus_TradeStatusCode] Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_D_TradeStatus_TradeStatusCode] ON [dbo].[D_TradeStatus]
```

```
(
    [TradeStatusCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING ON
```

GO

/\*\*\*\*\* Object: Index [IDX\_D\_TradeType\_TradeTypeCode] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IDX\_D\_TradeType\_TradeTypeCode] ON [dbo].[D\_TradeType]

(  
    [TradeTypeCode] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]  
GO

/\*\*\*\*\* Object: Index [IDX\_F\_DailyMarketPerformance\_Company\_SK] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IDX\_F\_DailyMarketPerformance\_Company\_SK] ON [dbo].[F\_DailyMarketPerformance]

(  
    [Company\_SK] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]  
GO

/\*\*\*\*\* Object: Index [IDX\_F\_DailyMarketPerformance\_Exchange\_SK] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IDX\_F\_DailyMarketPerformance\_Exchange\_SK] ON [dbo].[F\_DailyMarketPerformance]

(  
    [Exchange\_SK] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]  
GO

/\*\*\*\*\* Object: Index [IDX\_F\_DailyMarketPerformance\_Security\_SK] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IDX\_F\_DailyMarketPerformance\_Security\_SK] ON [dbo].[F\_DailyMarketPerformance]

(  
    [Security\_SK] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]  
GO

/\*\*\*\*\* Object: Index [IDX\_F\_QuarterlyCompanyPerformance\_Company\_SK] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IDX\_F\_QuarterlyCompanyPerformance\_Company\_SK] ON [dbo].[F\_QuarterlyCompanyPerformance]

(  
    [Company\_SK] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]

GO

```
/****** Object: Index [IDX_F_QuarterlyCompanyPerformance_FiscalQuarterStartDate_SK] Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_F_QuarterlyCompanyPerformance_FiscalQuarterStartDate_SK] ON [dbo].[F_QuarterlyCompanyPerformance]
(
    [FiscalQuarterStartDate_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/****** Object: Index [IDX_F_Trade_Broker_SK] Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_F_Trade_Broker_SK] ON [dbo].[F_Trade]
(
    [Broker_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/****** Object: Index [IDX_F_Trade_Company_SK] Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_F_Trade_Company_SK] ON [dbo].[F_Trade]
(
    [Company_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/****** Object: Index [IDX_F_Trade_CustomerAccount_SK] Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_F_Trade_CustomerAccount_SK] ON [dbo].[F_Trade]
(
    [CustomerAccount_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/****** Object: Index [IDX_F_Trade_Exchange_SK] Script Date: 2013-02-13 06:14:00 AM *****/
CREATE NONCLUSTERED INDEX [IDX_F_Trade_Exchange_SK] ON [dbo].[F_Trade]
(
    [Exchange_SK] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF,
ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
```

GO

/\*\*\*\*\* Object: Index [IDX\_F\_Trade\_Security\_SK] Script Date: 2013-02-13 06:14:00 AM \*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IDX\_F\_Trade\_Security\_SK] ON [dbo].[F\_Trade]

(  
    [Security\_SK] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = OFF, ALLOW\_PAGE\_LOCKS = OFF, DATA\_COMPRESSION = PAGE) ON [PRIMARY]  
GO

ALTER TABLE [dbo].[F\_DailyMarketPerformance] WITH NOCHECK ADD CONSTRAINT [FK\_F\_MarketPerformance\_D\_Company] FOREIGN  
KEY([Company\_SK])  
REFERENCES [dbo].[D\_Company] ([Company\_SK])  
NOT FOR REPLICATION  
GO

ALTER TABLE [dbo].[F\_DailyMarketPerformance] NOCHECK CONSTRAINT [FK\_F\_MarketPerformance\_D\_Company]  
GO

ALTER TABLE [dbo].[F\_DailyMarketPerformance] WITH NOCHECK ADD CONSTRAINT [FK\_F\_MarketPerformance\_D\_Date] FOREIGN KEY([Date\_SK])  
REFERENCES [dbo].[D\_Date] ([Date\_SK])  
NOT FOR REPLICATION  
GO

ALTER TABLE [dbo].[F\_DailyMarketPerformance] NOCHECK CONSTRAINT [FK\_F\_MarketPerformance\_D\_Date]  
GO

ALTER TABLE [dbo].[F\_DailyMarketPerformance] WITH NOCHECK ADD CONSTRAINT [FK\_F\_MarketPerformance\_D\_Exchange] FOREIGN  
KEY([Exchange\_SK])  
REFERENCES [dbo].[D\_Exchange] ([Exchange\_SK])  
NOT FOR REPLICATION  
GO

ALTER TABLE [dbo].[F\_DailyMarketPerformance] NOCHECK CONSTRAINT [FK\_F\_MarketPerformance\_D\_Exchange]  
GO

ALTER TABLE [dbo].[F\_DailyMarketPerformance] WITH NOCHECK ADD CONSTRAINT [FK\_F\_MarketPerformance\_D\_Security] FOREIGN  
KEY([Security\_SK])  
REFERENCES [dbo].[D\_Security] ([Security\_SK])  
NOT FOR REPLICATION

GO

```
ALTER TABLE [dbo].[F_DailyMarketPerformance] NOCHECK CONSTRAINT [FK_F_MarketPerformance_D_Security]
GO
```

```
ALTER TABLE [dbo].[F_QuarterlyCompanyPerformance] WITH NOCHECK ADD CONSTRAINT [FK_F_CompanyPerformance_D_Company] FOREIGN
KEY([Company_SK])
REFERENCES [dbo].[D_Company] ([Company_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_QuarterlyCompanyPerformance] NOCHECK CONSTRAINT [FK_F_CompanyPerformance_D_Company]
GO
```

```
ALTER TABLE [dbo].[F_QuarterlyCompanyPerformance] WITH NOCHECK ADD CONSTRAINT [FK_F_CompanyPerformance_D_Quarter] FOREIGN
KEY([FiscalQuarter_SK])
REFERENCES [dbo].[D_FiscalQuarter] ([FiscalQuarter_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_QuarterlyCompanyPerformance] NOCHECK CONSTRAINT [FK_F_CompanyPerformance_D_Quarter]
GO
```

```
ALTER TABLE [dbo].[F_QuarterlyCompanyPerformance] WITH NOCHECK ADD CONSTRAINT [FK_F_QuarterlyCompanyPerformance_D_Date] FOREIGN
KEY([FiscalQuarterStartDate_SK])
REFERENCES [dbo].[D_Date] ([Date_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_QuarterlyCompanyPerformance] NOCHECK CONSTRAINT [FK_F_QuarterlyCompanyPerformance_D_Date]
GO
```

```
ALTER TABLE [dbo].[F_Trade] WITH NOCHECK ADD CONSTRAINT [FK_F_Trade_D_Broker] FOREIGN KEY([Broker_SK])
REFERENCES [dbo].[D_Broker] ([Broker_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_Trade] NOCHECK CONSTRAINT [FK_F_Trade_D_Broker]
GO
```

```
ALTER TABLE [dbo].[F_Trade] WITH NOCHECK ADD CONSTRAINT [FK_F_Trade_D_CashType] FOREIGN KEY([CashType_SK])
REFERENCES [dbo].[D_CashType] ([CashType_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_Trade] NOCHECK CONSTRAINT [FK_F_Trade_D_CashType]
GO
```

```
ALTER TABLE [dbo].[F_Trade] WITH NOCHECK ADD CONSTRAINT [FK_F_Trade_D_Company] FOREIGN KEY([Company_SK])
REFERENCES [dbo].[D_Company] ([Company_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_Trade] NOCHECK CONSTRAINT [FK_F_Trade_D_Company]
GO
```

```
ALTER TABLE [dbo].[F_Trade] WITH NOCHECK ADD CONSTRAINT [FK_F_Trade_D_CustomerAccount] FOREIGN KEY([CustomerAccount_SK])
REFERENCES [dbo].[D_CustomerAccount] ([CustomerAccount_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_Trade] NOCHECK CONSTRAINT [FK_F_Trade_D_CustomerAccount]
GO
```

```
ALTER TABLE [dbo].[F_Trade] WITH NOCHECK ADD CONSTRAINT [FK_F_Trade_D_Date] FOREIGN KEY([Date_SK])
REFERENCES [dbo].[D_Date] ([Date_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_Trade] NOCHECK CONSTRAINT [FK_F_Trade_D_Date]
GO
```

```
ALTER TABLE [dbo].[F_Trade] WITH NOCHECK ADD CONSTRAINT [FK_F_Trade_D_Exchange] FOREIGN KEY([Exchange_SK])
REFERENCES [dbo].[D_Exchange] ([Exchange_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_Trade] NOCHECK CONSTRAINT [FK_F_Trade_D_Exchange]
GO
```

```
ALTER TABLE [dbo].[F_Trade] WITH NOCHECK ADD CONSTRAINT [FK_F_Trade_D_Security] FOREIGN KEY([Security_SK])
REFERENCES [dbo].[D_Security] ([Security_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_Trade] NOCHECK CONSTRAINT [FK_F_Trade_D_Security]
GO
```

```
ALTER TABLE [dbo].[F_Trade] WITH NOCHECK ADD CONSTRAINT [FK_F_Trade_D_TradeStatus] FOREIGN KEY([TradeStatus_SK])
REFERENCES [dbo].[D_TradeStatus] ([TradeStatus_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_Trade] NOCHECK CONSTRAINT [FK_F_Trade_D_TradeStatus]
GO
```

```
ALTER TABLE [dbo].[F_Trade] WITH NOCHECK ADD CONSTRAINT [FK_F_Trade_D_TradeType] FOREIGN KEY([TradeType_SK])
REFERENCES [dbo].[D_TradeType] ([TradeType_SK])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[F_Trade] NOCHECK CONSTRAINT [FK_F_Trade_D_TradeType]
GO
```

```
USE [master]
GO
```

```
ALTER DATABASE [TPCE_DM] SET READ_WRITE
GO
```

### 1.1.3 TPC\_DV Create Script

```
USE [master]
GO
```

```
/***** Object: Database [TPCE_DV] Script Date: 11/30/2011 10:57:19 *****/
```



```
CREATE DATABASE [TPCE_DV] ON PRIMARY
( NAME = N'TPCE_DV', FILENAME = N'D:\SQLDataFiles\MSSQL10.MSSQLSERVER\MSSQL\DATA\TPCE_DV.mdf' , SIZE = 6291456KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 1024KB )
LOG ON
( NAME = N'TPCE_DV_log', FILENAME = N'C:\SQLLogFiles\MSSQL10.MSSQLSERVER\MSSQL\DATA\TPCE_DV_log.ldf' , SIZE = 10600832KB , MAXSIZE =
2048GB , FILEGROWTH = 10%)
GO

ALTER DATABASE [TPCE_DV] SET COMPATIBILITY_LEVEL = 100
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
BEGIN
EXEC [TPCE_DV].[dbo].[sp_fulltext_database] @action = 'enable'
END
GO

ALTER DATABASE [TPCE_DV] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [TPCE_DV] SET ANSI_NULLS OFF
GO

ALTER DATABASE [TPCE_DV] SET ANSI_PADDING OFF
GO

ALTER DATABASE [TPCE_DV] SET ANSI_WARNINGS OFF
GO

ALTER DATABASE [TPCE_DV] SET ARITHABORT OFF
GO

ALTER DATABASE [TPCE_DV] SET AUTO_CLOSE OFF
GO

ALTER DATABASE [TPCE_DV] SET AUTO_CREATE_STATISTICS ON
GO

ALTER DATABASE [TPCE_DV] SET AUTO_SHRINK OFF
GO
```

```
ALTER DATABASE [TPCE_DV] SET AUTO_UPDATE_STATISTICS ON  
GO
```

```
ALTER DATABASE [TPCE_DV] SET CURSOR_CLOSE_ON_COMMIT OFF  
GO
```

```
ALTER DATABASE [TPCE_DV] SET CURSOR_DEFAULT GLOBAL  
GO
```

```
ALTER DATABASE [TPCE_DV] SET CONCAT_NULL_YIELDS_NULL OFF  
GO
```

```
ALTER DATABASE [TPCE_DV] SET NUMERIC_ROUNDABORT OFF  
GO
```

```
ALTER DATABASE [TPCE_DV] SET QUOTED_IDENTIFIER OFF  
GO
```

```
ALTER DATABASE [TPCE_DV] SET RECURSIVE_TRIGGERS OFF  
GO
```

```
ALTER DATABASE [TPCE_DV] SET DISABLE_BROKER  
GO
```

```
ALTER DATABASE [TPCE_DV] SET AUTO_UPDATE_STATISTICS_ASYNC OFF  
GO
```

```
ALTER DATABASE [TPCE_DV] SET DATE_CORRELATION_OPTIMIZATION OFF  
GO
```

```
ALTER DATABASE [TPCE_DV] SET TRUSTWORTHY OFF  
GO
```

```
ALTER DATABASE [TPCE_DV] SET ALLOW_SNAPSHOT_ISOLATION OFF  
GO
```

```
ALTER DATABASE [TPCE_DV] SET PARAMETERIZATION SIMPLE  
GO
```

```
ALTER DATABASE [TPCE_DV] SET READ_COMMITTED_SNAPSHOT OFF
GO
```

```
ALTER DATABASE [TPCE_DV] SET HONOR_BROKER_PRIORITY OFF
GO
```

```
ALTER DATABASE [TPCE_DV] SET READ_WRITE
GO
```

```
ALTER DATABASE [TPCE_DV] SET RECOVERY SIMPLE
GO
```

```
ALTER DATABASE [TPCE_DV] SET MULTI_USER
GO
```

```
ALTER DATABASE [TPCE_DV] SET PAGE_VERIFY CHECKSUM
GO
```

```
ALTER DATABASE [TPCE_DV] SET DB_CHAINING OFF
GO
```

```
EXEC sys.sp_db_vardecimal_storage_format N'TPCE_DV', N'ON'
GO
```

```
USE [TPCE_DV]
GO
```

```
/***** Object: Table [dbo].[HUB_TradeType] Script Date: 11/30/2011 10:57:20 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[HUB_TradeType](
    [HUB_TradeType_SQN] [int] IDENTITY(1,1) NOT NULL,
    [HUB_TradeType_LDTS] [datetime2](0) NOT NULL,
```

```

        [HUB_TradeType_RSRC] [varchar](10) NOT NULL,
        [TradeTypeDesc] [varchar](12) NOT NULL,
        CONSTRAINT [PK_HUB_TradeType] PRIMARY KEY CLUSTERED
    (
        [HUB_TradeType_SQN] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
    DATA_COMPRESSION = PAGE) ON [PRIMARY]
    ) ON [PRIMARY]
GO

SET ANSI_PADDING OFF
GO

CREATE UNIQUE NONCLUSTERED INDEX [IDX_HUB_TradeType_TradeTypeDesc] ON [dbo].[HUB_TradeType]
(
    [TradeTypeDesc] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Table [dbo].[HUB_Security]    Script Date: 11/30/2011 10:57:20 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[HUB_Security](
    [HUB_Security_SQN] [int] IDENTITY(1,1) NOT NULL,
    [HUB_Security_LDTS] [datetime2](0) NOT NULL,
    [HUB_Security_RSRC] [varchar](10) NOT NULL,
    [SecurityCode] [varchar](15) NOT NULL,
    CONSTRAINT [PK_HUB_Security] PRIMARY KEY CLUSTERED
    (
        [HUB_Security_SQN] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
    DATA_COMPRESSION = PAGE) ON [PRIMARY]

```

```
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
CREATE UNIQUE NONCLUSTERED INDEX [IDX_HUB_Security_SecurityCode] ON [dbo].[HUB_Security]
(
    [SecurityCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/****** Object: Table [dbo].[HUB_Exchange]    Script Date: 11/30/2011 10:57:20 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[HUB_Exchange](
    [HUB_Exchange_SQN] [int] IDENTITY(1,1) NOT NULL,
    [HUB_Exchange_LDTS] [datetime2](0) NOT NULL,
    [HUB_Exchange_RSRC] [varchar](10) NOT NULL,
    [ExchangeID] [varchar](6) NOT NULL,
    CONSTRAINT [PK_HUB_Exchange] PRIMARY KEY CLUSTERED
(
    [HUB_Exchange_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
CREATE UNIQUE NONCLUSTERED INDEX [IDX_HUB_Exchange_ExchangeID] ON [dbo].[HUB_Exchange]
```

```

(
    [ExchangeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Table [dbo].[HUB_Customer]    Script Date: 11/30/2011 10:57:20 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[HUB_Customer](
    [HUB_Customer_SQN] [int] IDENTITY(1,1) NOT NULL,
    [HUB_Customer_LDTS] [datetime2](0) NOT NULL,
    [HUB_Customer_RSRC] [varchar](10) NOT NULL,
    [CustomerID] [bigint] NOT NULL,
    CONSTRAINT [PK_HUB_Customer] PRIMARY KEY CLUSTERED
(
    [HUB_Customer_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_PADDING OFF
GO

CREATE UNIQUE NONCLUSTERED INDEX [IDX_HUB_Customer_CustomerID] ON [dbo].[HUB_Customer]
(
    [CustomerID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Table [dbo].[HUB_Company]    Script Date: 11/30/2011 10:57:21 *****/

```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[HUB_Company](
    [HUB_Company_SQN] [int] IDENTITY(1,1) NOT NULL,
    [HUB_Company_LDTS] [datetime2](0) NOT NULL,
    [HUB_Company_RSRC] [varchar](10) NOT NULL,
    [CompanyName] [varchar](60) NOT NULL,
    CONSTRAINT [PK_HUB_Company] PRIMARY KEY CLUSTERED
(
    [HUB_Company_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
CREATE UNIQUE NONCLUSTERED INDEX [IDX_HUB_Company_CompanyName] ON [dbo].[HUB_Company]
(
    [CompanyName] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/***** Object: Table [dbo].[HUB_Broker]    Script Date: 11/30/2011 10:57:21 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
```

GO

```
CREATE TABLE [dbo].[HUB_Broker](
    [HUB_Broker_SQN] [int] IDENTITY(1,1) NOT NULL,
    [HUB_Broker_LDTS] [datetime2](0) NOT NULL,
    [HUB_Broker_RSRC] [varchar](10) NOT NULL,
    [BrokerID] [bigint] NOT NULL,
    CONSTRAINT [PK_HUB_Broker] PRIMARY KEY CLUSTERED
(
    [HUB_Broker_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
CREATE UNIQUE NONCLUSTERED INDEX [IDX_HUB_Broker_BrokerID] ON [dbo].[HUB_Broker]
(
    [BrokerID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/***** Object: Table [dbo].[HUB_Account]    Script Date: 11/30/2011 10:57:21 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[HUB_Account](
    [HUB_Account_SQN] [int] IDENTITY(1,1) NOT NULL,
    [HUB_Account_LDTS] [datetime2](0) NOT NULL,
    [HUB_Account_RSRC] [varchar](10) NOT NULL,
    [AccountID] [bigint] NOT NULL,
```



```

CONSTRAINT [PK_HUB_Account] PRIMARY KEY CLUSTERED
(
    [HUB_Account_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_PADDING OFF
GO

CREATE UNIQUE NONCLUSTERED INDEX [IDX_HUB_Account_AccountID] ON [dbo].[HUB_Account]
(
    [AccountID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Table [dbo].[XREF_Town]    Script Date: 11/30/2011 10:57:21 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[XREF_Town](
    [PostalCode] [varchar](12) NOT NULL,
    [XREF_Town_LDTS] [datetime2](0) NOT NULL,
    [XREF_Town_RSRC] [varchar](10) NOT NULL,
    [TownName] [varchar](80) NOT NULL,
    [StateProvinceCode] [varchar](80) NOT NULL,
    [CountryName] [varchar](80) NOT NULL,
    CONSTRAINT [PK_XREF_Town] PRIMARY KEY CLUSTERED
(
    [PostalCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]

```

```
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[XREF_Status]    Script Date: 11/30/2011 10:57:21 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[XREF_Status](
    [StatusCode] [varchar](4) NOT NULL,
    [XREF_Status_LDTS] [datetime2](0) NOT NULL,
    [XREF_Status_RSRC] [varchar](10) NOT NULL,
    [StatusDesc] [varchar](10) NOT NULL,
    CONSTRAINT [PK_XREF_Status] PRIMARY KEY CLUSTERED
(
    [StatusCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[XREF_Sector]    Script Date: 11/30/2011 10:57:21 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
```

GO

```
CREATE TABLE [dbo].[XREF_Sector](
    [SectorCode] [varchar](2) NOT NULL,
    [XREF_Sector_LDTS] [datetime2](0) NOT NULL,
    [XREF_Sector_RSRC] [varchar](10) NOT NULL,
    [SectorDesc] [varchar](30) NOT NULL,
    CONSTRAINT [PK_XREF_Sector] PRIMARY KEY CLUSTERED
(
    [SectorCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/****** Object: Table [dbo].[XREF_Industry]    Script Date: 11/30/2011 10:57:21 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[XREF_Industry](
    [IndustryCode] [varchar](2) NOT NULL,
    [XREF_Industry_LDTS] [datetime2](0) NOT NULL,
    [XREF_Industry_RSRC] [varchar](10) NOT NULL,
    [IndustryDesc] [varchar](50) NOT NULL,
    CONSTRAINT [PK_XREF_Industry] PRIMARY KEY CLUSTERED
(
    [IndustryCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: View [dbo].[V_XREF_Town]    Script Date: 11/30/2011 10:57:21 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE VIEW [dbo].[V_XREF_Town]
AS
SELECT DISTINCT
    zc.ZC_CODE AS PostalCode,
    zc.ZC_TOWN AS TownName,
    zc.ZC_DIV AS StateProvinceCode,
    ad.AD_CTRY AS CountryName
FROM
    TPCE.dbo.[ADDRESS] ad WITH (NOLOCK)
    JOIN TPCE.dbo.ZIP_CODE zc WITH (NOLOCK)
        ON zc.ZC_CODE = ad.AD_ZC_CODE;
GO
```

```
/***** Object: View [dbo].[V_XREF_Status]  Script Date: 11/30/2011 10:57:21 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE VIEW [dbo].[V_XREF_Status]
AS
SELECT
    st.ST_ID AS StatusCode,
    st.ST_NAME AS StatusDesc
FROM
    TPCE.dbo.STATUS_TYPE st WITH (NOLOCK);
GO
```

```
/***** Object: View [dbo].[V_XREF_Sector]    Script Date: 11/30/2011 10:57:21 *****/  
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
CREATE VIEW [dbo].[V_XREF_Sector]  
AS  
SELECT  
    sc.SC_ID AS SectorCode,  
    sc.SC_NAME AS SectorDesc  
FROM  
    TPCE.dbo.SECTOR sc WITH (NOLOCK);  
GO
```

```
/***** Object: View [dbo].[V_XREF_Industry]    Script Date: 11/30/2011 10:57:21 *****/  
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
CREATE VIEW [dbo].[V_XREF_Industry]  
AS  
SELECT  
    ind.IN_ID AS IndustryCode,  
    ind.IN_NAME AS IndustryDesc  
FROM  
    TPCE.dbo.INDUSTRY ind WITH (NOLOCK);  
GO
```

```
/***** Object: View [dbo].[V_SAT_TradeType]    Script Date: 11/30/2011 10:57:21 *****/  
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
CREATE VIEW [dbo].[V_SAT_TradeType]
```

```
AS
SELECT
    htt.HUB_TradeType_SQN AS HUB_TradeType_SQN,
    tt.TT_ID AS TradeTypeCode,
    tt.TT_IS_SELL AS IsSellFlag,
    tt.TT_IS_MRKT AS IsMarketFlag
FROM
    TPCE.dbo.TRADE_TYPE tt WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_TradeType htt WITH (NOLOCK)
        ON htt.TradeTypeDesc = tt.TT_NAME;
GO
```

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_CompanyCreditRating] Script Date: 11/30/2011 10:57:21 \*\*\*\*\*/

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE VIEW [dbo].[V_SAT_CompanyCreditRating]
AS
SELECT
    hco.HUB_Company_SQN AS HUB_Company_SQN,
    co.CO_SP_RATE AS CompanyCreditRating
FROM
    TPCE.dbo.COMPANY co WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
        ON hco.CompanyName = co.CO_NAME;
GO
```

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_CompanyCEO] Script Date: 11/30/2011 10:57:21 \*\*\*\*\*/

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE VIEW [dbo].[V_SAT_CompanyCEO]
AS
SELECT
```

```
        hco.HUB_Company_SQN AS HUB_Company_SQN,  
        co.CO_CEO AS CompanyCEO  
FROM  
    TPCE.dbo.COMPANY co WITH (NOLOCK)  
    JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)  
        ON hco.CompanyName = co.CO_NAME;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_Company] Script Date: 11/30/2011 10:57:21 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_SAT\_Company]

AS

SELECT

```
    hco.HUB_Company_SQN AS HUB_Company_SQN,  
    co.CO_ID AS CompanyID,  
    co.CO_DESC AS CompanyDesc,  
    co.CO_OPEN_DATE AS CompanyOpenDate,  
    ad.AD_LINE1 AS AddressLine1,  
    ad.AD_LINE2 AS AddressLine2,  
    ad.AD_ZC_CODE AS PostalCode,  
    ind.IN_ID AS IndustryCode,  
    ind.IN_SC_ID AS SectorCode,  
    co.CO_ST_ID AS StatusCode
```

FROM

```
    TPCE.dbo.COMPANY co WITH (NOLOCK)  
    JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)  
        ON hco.CompanyName = co.CO_NAME  
    JOIN TPCE.dbo.[ADDRESS] ad WITH (NOLOCK)  
        ON ad.AD_ID = co.CO_AD_ID  
    JOIN TPCE.dbo.INDUSTRY ind WITH (NOLOCK)  
        ON ind.IN_ID = co.CO_IN_ID;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_Broker] Script Date: 11/30/2011 10:57:21 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_SAT\_Broker]

AS

SELECT

hb.HUB\_Broker\_SQN AS HUB\_Broker\_SQN,  
b.B\_NAME AS BrokerName,  
b.B\_ST\_ID AS StatusCode

FROM

TPCE.dbo.[BROKER] b WITH (NOLOCK)  
JOIN TPCE\_DV.dbo.HUB\_Broker hb WITH (NOLOCK)  
ON hb.BrokerID = b.B\_ID;

GO

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_Security] Script Date: 11/30/2011 10:57:21 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_SAT\_Security]

AS

SELECT

hs.HUB\_Security\_SQN AS HUB\_Security\_SQN,  
s.S\_NAME AS SecurityName,  
s.S\_ISSUE AS IssueType,  
s.S\_START\_DATE AS FirstTradeDate,  
s.S\_ST\_ID AS StatusCode

FROM

TPCE.dbo.[SECURITY] s WITH (NOLOCK)  
JOIN TPCE\_DV.dbo.HUB\_Security hs WITH (NOLOCK)  
ON hs.SecurityCode = s.S\_SYMB;

GO

/\*\*\*\*\* Object: Table [dbo].[SAT\_Account] Script Date: 11/30/2011 10:57:21 \*\*\*\*\*/

SET ANSI\_NULLS ON



GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[SAT_Account](
    [HUB_Account_SQN] [int] NOT NULL,
    [SAT_Account_LDTS] [datetime2](0) NOT NULL,
    [SAT_Account_LEDTS] [datetime2](0) NULL,
    [SAT_Account_RSRC] [varchar](10) NULL,
    [AccountName] [varchar](50) NOT NULL,
    [TaxStatusCode] [tinyint] NOT NULL,
    CONSTRAINT [PK_SAT_Account] PRIMARY KEY CLUSTERED
(
    [HUB_Account_SQN] ASC,
    [SAT_Account_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[LNK\_SecurityExchange] Script Date: 11/30/2011 10:57:21 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[LNK_SecurityExchange](
    [LNK_SecurityExchange_SQN] [int] IDENTITY(1,1) NOT NULL,
    [LNK_SecurityExchange_LDTS] [datetime2](0) NOT NULL,
```

```

        [LNK_SecurityExchange_RSRC] [varchar](10) NOT NULL,
        [HUB_Security_SQN] [int] NOT NULL,
        [HUB_Exchange_SQN] [int] NOT NULL,
    CONSTRAINT [PK_LNK_SecurityExchange] PRIMARY KEY CLUSTERED
    (
        [LNK_SecurityExchange_SQN] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
    DATA_COMPRESSION = PAGE) ON [PRIMARY]
    ) ON [PRIMARY]
GO

SET ANSI_PADDING OFF
GO

CREATE NONCLUSTERED INDEX [IDX_LNK_SecurityExchange_HUB_Security_SQN] ON [dbo].[LNK_SecurityExchange]
(
    [HUB_Security_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Table [dbo].[LNK_SecurityCompany]    Script Date: 11/30/2011 10:57:21 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[LNK_SecurityCompany](
    [LNK_SecurityCompany_SQN] [int] IDENTITY(1,1) NOT NULL,
    [LNK_SecurityCompany_LDTS] [datetime2](0) NOT NULL,
    [LNK_SecurityCompany_RSRC] [varchar](10) NOT NULL,
    [HUB_Security_SQN] [int] NOT NULL,
    [HUB_Company_SQN] [int] NOT NULL,
    CONSTRAINT [PK_LNK_SecurityCompany] PRIMARY KEY CLUSTERED
    (
        [LNK_SecurityCompany_SQN] ASC
    )

```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
CREATE NONCLUSTERED INDEX [IDX_LNK_SecurityCompany_HUB_Company_SQN] ON [dbo].[LNK_SecurityCompany]
(
    [HUB_Company_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
CREATE NONCLUSTERED INDEX [IDX_LNK_SecurityCompany_HUB_Security_SQN] ON [dbo].[LNK_SecurityCompany]
(
    [HUB_Security_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/****** Object: Table [dbo].[LNK_AccountCustomer]    Script Date: 11/30/2011 10:57:21 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[LNK_AccountCustomer](
    [LNK_AccountCustomer_SQN] [int] IDENTITY(1,1) NOT NULL,
    [LNK_AccountCustomer_LDTS] [datetime2](0) NOT NULL,
    [LNK_AccountCustomer_RSRC] [varchar](10) NOT NULL,
    [HUB_Account_SQN] [int] NOT NULL,
    [HUB_Customer_SQN] [int] NOT NULL,
    CONSTRAINT [PK_LNK_AccountCustomer] PRIMARY KEY CLUSTERED
(
```

```
        [LNK_AccountCustomer_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
CREATE NONCLUSTERED INDEX [IDX_LNK_AccountCustomer_HUB_Account_SQN] ON [dbo].[LNK_AccountCustomer]
(
    [HUB_Account_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
CREATE NONCLUSTERED INDEX [IDX_LNK_AccountCustomer_HUB_Customer_SQN] ON [dbo].[LNK_AccountCustomer]
(
    [HUB_Customer_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/***** Object: Table [dbo].[LNK_AccountBroker]    Script Date: 11/30/2011 10:57:21 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[LNK_AccountBroker](
    [LNK_AccountBroker_SQN] [int] IDENTITY(1,1) NOT NULL,
    [LNK_AccountBroker_LDTS] [datetime2](0) NOT NULL,
    [LNK_AccountBroker_RSRC] [varchar](10) NOT NULL,
    [HUB_Account_SQN] [int] NOT NULL,
    [HUB_Broker_SQN] [int] NOT NULL,
    CONSTRAINT [PK_LNK_AccountBroker] PRIMARY KEY CLUSTERED
```

```
(
    [LNK_AccountBroker_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
CREATE NONCLUSTERED INDEX [IDX_LNK_AccountBroker_HUB_Account_SQN] ON [dbo].[LNK_AccountBroker]
(
    [HUB_Account_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
CREATE NONCLUSTERED INDEX [IDX_LNK_AccountBroker_HUB_Broker_SQN] ON [dbo].[LNK_AccountBroker]
(
    [HUB_Broker_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/***** Object: Table [dbo].[TLNK_Trade]    Script Date: 11/30/2011 10:57:21 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[TLNK_Trade](
    [TradeID] [bigint] NOT NULL,
    [TLNK_Trade_LDTS] [datetime2](0) NOT NULL,
    [TLNK_Trade_RSRC] [varchar](10) NOT NULL,
    [HUB_Broker_SQN] [int] NOT NULL,
    [HUB_Customer_SQN] [int] NOT NULL,
```

```

[HUB_Account_SQN] [int] NOT NULL,
[HUB_Company_SQN] [int] NOT NULL,
[HUB_Exchange_SQN] [int] NOT NULL,
[HUB_Security_SQN] [int] NOT NULL,
[HUB_TradeType_SQN] [int] NOT NULL,
[TradeDateTime] [datetime2](0) NOT NULL,
[Quantity] [int] NOT NULL,
[BidPrice] [money] NOT NULL,
[TradePrice] [money] NOT NULL,
[Charge] [money] NOT NULL,
[Commission] [money] NOT NULL,
[Tax] [money] NOT NULL,
[IsCashFlag] [bit] NOT NULL,
[StatusCode] [varchar](4) NOT NULL,
CONSTRAINT [PK_TLNK_Trade] PRIMARY KEY CLUSTERED
(
    [TradeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_PADDING OFF
GO

CREATE NONCLUSTERED INDEX [IDX_TLNK_Trade_HUB_Account_SQN] ON [dbo].[TLNK_Trade]
(
    [HUB_Account_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

CREATE NONCLUSTERED INDEX [IDX_TLNK_Trade_HUB_Broker_SQN] ON [dbo].[TLNK_Trade]
(
    [HUB_Broker_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

CREATE NONCLUSTERED INDEX [IDX_TLNK_Trade_HUB_Company_SQN] ON [dbo].[TLNK_Trade]

```

```
(
    [HUB_Company_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
CREATE NONCLUSTERED INDEX [IDX_TLNK_Trade_HUB_Customer_SQN] ON [dbo].[TLNK_Trade]
```

```
(
    [HUB_Customer_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
CREATE NONCLUSTERED INDEX [IDX_TLNK_Trade_HUB_Security_SQN] ON [dbo].[TLNK_Trade]
```

```
(
    [HUB_Security_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/****** Object: Table [dbo].[TLNK_QuarterlyCompanyPerformance]    Script Date: 11/30/2011 10:57:21 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[TLNK_QuarterlyCompanyPerformance](
    [HUB_Company_SQN] [int] NOT NULL,
    [FiscalQuarter] [int] NOT NULL,
    [FiscalYear] [int] NOT NULL,
    [TLNK_QuarterlyCompanyPerformance_LDTS] [datetime2](0) NOT NULL,
    [TLNK_QuarterlyCompanyPerformance_RSRC] [varchar](10) NOT NULL,
    [FiscalQuarterStartDate] [date] NOT NULL,
    [Revenue] [money] NOT NULL,
    [NetEarnings] [money] NOT NULL,
    [EarningsPerShareBasic] [money] NOT NULL,
```

```

[EarningsPerShareDiluted] [money] NOT NULL,
[Margin] [money] NOT NULL,
[Inventory] [money] NOT NULL,
[Assets] [money] NOT NULL,
[Liabilities] [money] NOT NULL,
[SharesOutstandingBasicAvg] [bigint] NOT NULL,
[SharesOutstandingDilutedAvg] [bigint] NOT NULL,
CONSTRAINT [PK_TLNK_CompanyPerformance] PRIMARY KEY CLUSTERED
(
    [FiscalQuarter] ASC,
    [FiscalYear] ASC,
    [HUB_Company_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[TLNK_DailyMarketPerformance]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[TLNK_DailyMarketPerformance](
    [HUB_Security_SQN] [int] NOT NULL,
    [Date] [date] NOT NULL,
    [TLNK_DailyMarketPerformance_LDTS] [datetime2](0) NOT NULL,
    [TLNK_DailyMarketPerformance_RSRC] [varchar](10) NOT NULL,
    [HUB_Company_SQN] [int] NOT NULL,
    [HUB_Exchange_SQN] [int] NOT NULL,
    [DayClose] [money] NOT NULL,
    [DayHigh] [money] NOT NULL,
    [DayLow] [money] NOT NULL,

```



```

        [DayVolume] [bigint] NOT NULL,
    CONSTRAINT [PK_TLNK_MarketPerformance] PRIMARY KEY CLUSTERED
(
    [Date] ASC,
    [HUB_Security_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

CREATE NONCLUSTERED INDEX [IDX_TLNK_DailyMarketPerformance_HUB_Company_SQN] ON [dbo].[TLNK_DailyMarketPerformance]
(
    [HUB_Company_SQN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

```

```

/***** Object: View [dbo].[V_HUB_Company]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

CREATE VIEW [dbo].[V_HUB_Company]
AS
SELECT
    co.CO_NAME CompanyName
FROM
    TPCE.dbo.COMPANY co WITH (NOLOCK);
GO

```

```

/***** Object: View [dbo].[V_HUB_Broker]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON

```

GO

CREATE VIEW [dbo].[V\_HUB\_Broker]

AS

SELECT

b.B\_ID BrokerID

FROM

TPCE.dbo.[BROKER] b WITH (NOLOCK);

GO

/\*\*\*\*\* Object: View [dbo].[V\_HUB\_Account] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_HUB\_Account]

AS

SELECT

ca.CA\_ID AS AccountID

FROM

TPCE.dbo.CUSTOMER\_ACCOUNT ca WITH (NOLOCK);

GO

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_Account] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_SAT\_Account]

AS

SELECT

ha.HUB\_Account\_SQN AS HUB\_Account\_SQN,

ca.CA\_NAME AS AccountName,

ca.CA\_TAX\_ST AS TaxStatusCode

FROM

TPCE.dbo.CUSTOMER\_ACCOUNT ca WITH (NOLOCK)

```
JOIN TPCE_DV.dbo.HUB_Account ha WITH (NOLOCK)
    ON ha.AccountID = ca.CA_ID;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_LNK\_SecurityExchange] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_LNK\_SecurityExchange]

AS

SELECT

```
    hs.HUB_Security_SQN AS HUB_Security_SQN,
    hex.HUB_Exchange_SQN AS HUB_Exchange_SQN
```

FROM

```
TPCE.dbo.[SECURITY] s WITH (NOLOCK)
```

```
JOIN TPCE_DV.dbo.HUB_Security hs WITH (NOLOCK)
```

```
    ON hs.SecurityCode = s.S_SYMB
```

```
JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)
```

```
    ON hex.ExchangeID = s.S_EX_ID;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_LNK\_SecurityCompany] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_LNK\_SecurityCompany]

AS

SELECT

```
    hs.HUB_Security_SQN AS HUB_Security_SQN,
```

```
    hco.HUB_Company_SQN AS HUB_Company_SQN
```

FROM

```
TPCE.dbo.[SECURITY] s WITH (NOLOCK)
```

```
JOIN TPCE_DV.dbo.HUB_Security hs WITH (NOLOCK)
```

```
    ON hs.SecurityCode = s.S_SYMB
```

```
JOIN TPCE.dbo.COMPANY co WITH (NOLOCK)
    ON co.CO_ID = s.S_CO_ID
JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
    ON hco.CompanyName = co.CO_NAME;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_LNK\_AccountCustomer] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_LNK\_AccountCustomer]

AS

SELECT

```
    ha.HUB_Account_SQN AS HUB_Account_SQN,
    hc.HUB_Customer_SQN AS HUB_Customer_SQN
```

FROM

```
    TPCE.dbo.CUSTOMER_ACCOUNT ca WITH (NOLOCK)
  JOIN TPCE_DV.dbo.HUB_Account ha WITH (NOLOCK)
      ON ha.AccountID = ca.CA_ID
  JOIN TPCE_DV.dbo.HUB_Customer hc WITH (NOLOCK)
      ON hc.CustomerID = ca.CA_C_ID;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_LNK\_AccountBroker] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_LNK\_AccountBroker]

AS

SELECT

```
    ha.HUB_Account_SQN AS HUB_Account_SQN,
    hb.HUB_Broker_SQN AS HUB_Broker_SQN
```

FROM

```
    TPCE.dbo.CUSTOMER_ACCOUNT ca WITH (NOLOCK)
```

```
JOIN TPCE_DV.dbo.HUB_Account ha WITH (NOLOCK)
    ON ha.AccountID = ca.CA_ID
JOIN TPCE_DV.dbo.HUB_Broker hb WITH (NOLOCK)
    ON hb.BrokerID = ca.CA_B_ID;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_HUB\_TradeType] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_HUB\_TradeType]

AS

SELECT

TT\_NAME AS TradeTypeDesc

FROM

TPCE.dbo.TRADE\_TYPE AS tt WITH (NOLOCK)

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F-00AA00A3EFFF, 1.00]

Begin DesignProperties =

Begin PaneConfigurations =

Begin PaneConfiguration = 0

NumPanels = 4

Configuration = "(H (1[40] 4[20] 2[20] 3) )"

End

Begin PaneConfiguration = 1

NumPanels = 3

Configuration = "(H (1 [50] 4 [25] 3))"

End

Begin PaneConfiguration = 2

NumPanels = 3

Configuration = "(H (1 [50] 2 [25] 3))"

End

Begin PaneConfiguration = 3

NumPanels = 3

Configuration = "(H (4 [30] 2 [40] 3))"

End

```
Begin PaneConfiguration = 4
    NumPanes = 2
    Configuration = "(H (1 [56] 3))"
End
Begin PaneConfiguration = 5
    NumPanes = 2
    Configuration = "(H (2 [66] 3))"
End
Begin PaneConfiguration = 6
    NumPanes = 2
    Configuration = "(H (4 [50] 3))"
End
Begin PaneConfiguration = 7
    NumPanes = 1
    Configuration = "(V (3))"
End
Begin PaneConfiguration = 8
    NumPanes = 3
    Configuration = "(H (1[56] 4[18] 2) )"
End
Begin PaneConfiguration = 9
    NumPanes = 2
    Configuration = "(H (1 [75] 4))"
End
Begin PaneConfiguration = 10
    NumPanes = 2
    Configuration = "(H (1[66] 2) )"
End
Begin PaneConfiguration = 11
    NumPanes = 2
    Configuration = "(H (4 [60] 2))"
End
Begin PaneConfiguration = 12
    NumPanes = 1
    Configuration = "(H (1) )"
End
Begin PaneConfiguration = 13
    NumPanes = 1
    Configuration = "(V (4))"
End
```

[illegible]

```

End
Begin CriteriaPane =
    Begin ColumnWidths = 11
        Column = 1440
        Alias = 2460
        Table = 1170
        Output = 720
        AppEND = 1400
        NewValue = 1170
        SortType = 1350
        SortOrder = 1410
        GroupBy = 1350
        Filter = 1350
        Or = 1350
        Or = 1350
        Or = 1350
    End
End
End
', @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'V_HUB_TradeType'
GO

EXEC sys.sp_addextENDEDproperty @name=N'MS_DiagramPaneCount', @value=1 , @level0type=N'SCHEMA',@level0name=N'dbo',
@level1type=N'VIEW',@level1name=N'V_HUB_TradeType'
GO

/***** Object: View [dbo].[V_HUB_Security]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[V_HUB_Security]
AS
SELECT
    s.S_SYMB SecurityCode
FROM
    TPCE.dbo.[SECURITY] s WITH (NOLOCK);
GO

```



/\*\*\*\*\* Object: Table [dbo].[SAT\_Security] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[SAT_Security](
    [HUB_Security_SQN] [int] NOT NULL,
    [SAT_Security_LDTS] [datetime2](0) NOT NULL,
    [SAT_Security_LEDTS] [datetime2](0) NULL,
    [SAT_Security_RSRC] [varchar](10) NOT NULL,
    [SecurityName] [varchar](70) NOT NULL,
    [IssueType] [varchar](6) NOT NULL,
    [FirstTradeDate] [date] NOT NULL,
    [StatusCode] [varchar](4) NOT NULL,
    CONSTRAINT [PK_SAT_Security] PRIMARY KEY CLUSTERED
(
    [HUB_Security_SQN] ASC,
    [SAT_Security_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[SAT\_ExchangeSecurities] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[SAT_ExchangeSecurities](
    [HUB_Exchange_SQN] [int] NOT NULL,
    [SAT_ExchangeSecurities_LDTS] [datetime2](0) NOT NULL,
    [SAT_ExchangeSecurities_LEDTS] [datetime2](0) NULL,
    [SAT_ExchangeSecurities_RSRC] [varchar](10) NOT NULL,
    [ListedSecurities] [int] NOT NULL,
    CONSTRAINT [PK_SAT_ExchangeSecurities] PRIMARY KEY CLUSTERED
(
    [HUB_Exchange_SQN] ASC,
    [SAT_ExchangeSecurities_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[SAT_Exchange]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[SAT_Exchange](
    [HUB_Exchange_SQN] [int] NOT NULL,
    [SAT_Exchange_LDTS] [datetime2](0) NOT NULL,
    [SAT_Exchange_LEDTS] [datetime2](0) NULL,
    [SAT_Exchange_RSRC] [varchar](10) NOT NULL,
    [ExchangeName] [varchar](100) NOT NULL,
    [ExchangeDesc] [varchar](150) NOT NULL,
    [OpeningTime] [smallint] NOT NULL,
    [ClosingTime] [smallint] NOT NULL,
    [AddressLine1] [varchar](80) NOT NULL,
```

```

        [AddressLine2] [varchar](80) NULL,
        [PostalCode] [varchar](12) NOT NULL,
CONSTRAINT [PK_SAT_Exchange] PRIMARY KEY CLUSTERED
(
        [HUB_Exchange_SQN] ASC,
        [SAT_Exchange_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_PADDING OFF
GO

CREATE NONCLUSTERED INDEX [IDX_SAT_Exchange_PostalCode] ON [dbo].[SAT_Exchange]
(
        [PostalCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Table [dbo].[SAT_CustomerContact]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[SAT_CustomerContact](
        [HUB_Customer_SQN] [int] NOT NULL,
        [SAT_CustomerContact_LDTS] [datetime2](0) NOT NULL,
        [SAT_CustomerContact_LEDTS] [datetime2](0) NULL,
        [SAT_CustomerContact_RSRC] [varchar](10) NOT NULL,
        [PhoneCountryCodePrimary] [varchar](3) NULL,
        [PhoneAreaCodePrimary] [varchar](3) NULL,
        [PhoneLocalNumberPrimary] [varchar](10) NULL,
        [PhoneExtensionPrimary] [varchar](5) NULL,

```

```

[PhoneCountryCodeSecondary] [varchar](3) NULL,
[PhoneAreaCodeSecondary] [varchar](3) NULL,
[PhoneLocalNumberSecondary] [varchar](10) NULL,
[PhoneExtensionSecondary] [varchar](5) NULL,
[EmailPrimary] [varchar](50) NULL,
[EmailSecondary] [varchar](50) NULL,
[AddressLine1] [varchar](80) NOT NULL,
[AddressLine2] [varchar](80) NULL,
[PostalCode] [varchar](12) NOT NULL,
CONSTRAINT [PK_SAT_CustomerContact] PRIMARY KEY CLUSTERED
(
    [HUB_Customer_SQN] ASC,
    [SAT_CustomerContact_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_PADDING OFF
GO

CREATE NONCLUSTERED INDEX [IDX_SAT_CustomerContact_PostalCode] ON [dbo].[SAT_CustomerContact]
(
    [PostalCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO

/***** Object: Table [dbo].[SAT_Customer]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[SAT_Customer](
    [HUB_Customer_SQN] [int] NOT NULL,

```

```

[SAT_Customer_LDTS] [datetime2](0) NOT NULL,
[SAT_Customer_LEDTS] [datetime2](0) NULL,
[SAT_Customer_RSRC] [varchar](10) NOT NULL,
[LastName] [varchar](25) NOT NULL,
[FirstName] [varchar](20) NOT NULL,
[MiddleInitial] [varchar](1) NULL,
[GENDER] [varchar](1) NULL,
[DateOfBirth] [date] NOT NULL,
[TierNumber] [tinyint] NOT NULL,
[StatusCode] [varchar](4) NOT NULL,
CONSTRAINT [PK_SAT_Customer] PRIMARY KEY CLUSTERED
(
    [HUB_Customer_SQN] ASC,
    [SAT_Customer_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[SAT_CompanyCreditRating]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[SAT_CompanyCreditRating](
    [HUB_Company_SQN] [int] NOT NULL,
    [SAT_CompanyCreditRating_LDTS] [datetime2](0) NOT NULL,
    [SAT_CompanyCreditRating_LEDTS] [datetime2](0) NULL,
    [SAT_CompanyCreditRating_RSRC] [varchar](10) NOT NULL,
    [CompanyCreditRating] [varchar](4) NOT NULL,
    CONSTRAINT [PK_SAT_CompanyCreditRating] PRIMARY KEY CLUSTERED
(

```

```
        [HUB_Company_SQN] ASC,  
        [SAT_CompanyCreditRating_LDTS] ASC  
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]  
GO
```

```
SET ANSI_PADDING OFF  
GO
```

```
/***** Object: Table [dbo].[SAT_CompanyCEO]    Script Date: 11/30/2011 10:57:22 *****/  
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
SET ANSI_PADDING ON  
GO
```

```
CREATE TABLE [dbo].[SAT_CompanyCEO](  
    [HUB_Company_SQN] [int] NOT NULL,  
    [SAT_CompanyCEO_LDTS] [datetime2](0) NOT NULL,  
    [SAT_CompanyCEO_LEDTS] [datetime2](0) NULL,  
    [SAT_CompanyCEO_RSRC] [varchar](10) NOT NULL,  
    [CompanyCEO] [varchar](46) NOT NULL,  
    CONSTRAINT [PK_SAT_CompanyCEO] PRIMARY KEY CLUSTERED  
(  
        [HUB_Company_SQN] ASC,  
        [SAT_CompanyCEO_LDTS] ASC  
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,  
DATA_COMPRESSION = PAGE) ON [PRIMARY]  
) ON [PRIMARY]  
GO
```

```
SET ANSI_PADDING OFF  
GO
```

```
/***** Object: Table [dbo].[SAT_Company]    Script Date: 11/30/2011 10:57:22 *****/  
SET ANSI_NULLS ON
```

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[SAT_Company](
    [HUB_Company_SQN] [int] NOT NULL,
    [SAT_Company_LDTS] [datetime2](0) NOT NULL,
    [SAT_Company_LEDTS] [datetime2](0) NULL,
    [SAT_Company_RSRC] [varchar](10) NOT NULL,
    [CompanyID] [bigint] NOT NULL,
    [CompanyDesc] [varchar](150) NOT NULL,
    [CompanyOpenDate] [date] NOT NULL,
    [AddressLine1] [varchar](80) NOT NULL,
    [AddressLine2] [varchar](80) NULL,
    [PostalCode] [varchar](12) NOT NULL,
    [IndustryCode] [varchar](2) NOT NULL,
    [SectorCode] [varchar](2) NOT NULL,
    [StatusCode] [varchar](4) NOT NULL,
    CONSTRAINT [PK_SAT_Company] PRIMARY KEY CLUSTERED
(
    [HUB_Company_SQN] ASC,
    [SAT_Company_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

SET ANSI_PADDING OFF
GO

CREATE NONCLUSTERED INDEX [IDX_SAT_Company_IndustryCode] ON [dbo].[SAT_Company]
(
    [IndustryCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE =
OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
CREATE NONCLUSTERED INDEX [IDX_SAT_Company_PostalCode] ON [dbo].[SAT_Company]
(
    [PostalCode] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
GO
```

```
/***** Object: Table [dbo].[SAT_Broker]    Script Date: 11/30/2011 10:57:22 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[SAT_Broker](
    [HUB_Broker_SQN] [int] NOT NULL,
    [SAT_Broker_LDTS] [datetime2](0) NOT NULL,
    [SAT_Broker_LEDTS] [datetime2](0) NULL,
    [SAT_Broker_RSRC] [varchar](10) NULL,
    [BrokerName] [varchar](49) NOT NULL,
    [StatusCode] [varchar](4) NOT NULL,
    CONSTRAINT [PK_SAT_Broker] PRIMARY KEY CLUSTERED
(
    [HUB_Broker_SQN] ASC,
    [SAT_Broker_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[SAT_TradeType]    Script Date: 11/30/2011 10:57:22 *****/
```

```
SET ANSI_NULLS ON
GO
```



```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[SAT_TradeType](
    [HUB_TradeType_SQN] [int] NOT NULL,
    [SAT_TradeType_LDTS] [datetime2](0) NOT NULL,
    [SAT_TradeType_LEDTS] [datetime2](0) NULL,
    [SAT_TradeType_RSRC] [varchar](10) NOT NULL,
    [TradeTypeCode] [varchar](3) NOT NULL,
    [IsSellFlag] [bit] NOT NULL,
    [IsMarketFlag] [bit] NOT NULL,
    CONSTRAINT [PK_SAT_TradeType] PRIMARY KEY CLUSTERED
(
    [HUB_TradeType_SQN] ASC,
    [SAT_TradeType_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
```

```
/***** Object: Table [dbo].[SAT_SecurityExchangeStatus]    Script Date: 11/30/2011 10:57:22 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[SAT_SecurityExchangeStatus](
    [LNK_SecurityExchange_SQN] [int] NOT NULL,
    [SAT_SecurityExchangeStatus_LDTS] [datetime2](0) NOT NULL,
```

```

        [SAT_SecurityExchangeStatus_LEDTS] [datetime2](0) NULL,
        [SAT_SecurityExchangeStatus_RSRC] [varchar](10) NOT NULL,
    CONSTRAINT [PK_SAT_SecurityExchangeStatus] PRIMARY KEY CLUSTERED
(
    [LNK_SecurityExchange_SQN] ASC,
    [SAT_SecurityExchangeStatus_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: Table [dbo].[SAT_SecurityCompanyStatus]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[SAT_SecurityCompanyStatus](
    [LNK_SecurityCompany_SQN] [int] NOT NULL,
    [SAT_SecurityCompanyStatus_LDTS] [datetime2](0) NOT NULL,
    [SAT_SecurityCompanyStatus_LEDTS] [datetime2](0) NULL,
    [SAT_SecurityCompanyStatus_RSRC] [varchar](10) NOT NULL,
    CONSTRAINT [PK_SAT_SecurityCompanyStatus] PRIMARY KEY CLUSTERED
(
    [LNK_SecurityCompany_SQN] ASC,
    [SAT_SecurityCompanyStatus_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

/\*\*\*\*\* Object: Table [dbo].[SAT\_AccountCustomerStatus] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[SAT_AccountCustomerStatus](
    [LNK_AccountCustomer_SQN] [int] NOT NULL,
    [SAT_AccountCustomerStatus_LDTS] [datetime2](0) NOT NULL,
    [SAT_AccountCustomerStatus_LEDTS] [datetime2](0) NULL,
    [SAT_AccountCustomerStatus_RSRC] [varchar](10) NOT NULL,
    CONSTRAINT [PK_SAT_AccountCustomerStatus] PRIMARY KEY CLUSTERED
(
    [LNK_AccountCustomer_SQN] ASC,
    [SAT_AccountCustomerStatus_LDTS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\* Object: Table [dbo].[SAT\_AccountBrokerStatus] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

```
CREATE TABLE [dbo].[SAT_AccountBrokerStatus](
    [LNK_AccountBroker_SQN] [int] NOT NULL,
```

```

        [SAT_AccountBrokerStatus_LDTS] [datetime2](0) NOT NULL,
        [SAT_AccountBrokerStatus_LEDTS] [datetime2](0) NULL,
        [SAT_AccountBrokerStatus_RSRC] [varchar](10) NOT NULL,
    CONSTRAINT [PK_SAT_AccountBrokerStatus] PRIMARY KEY CLUSTERED
    (
        [LNK_AccountBroker_SQN] ASC,
        [SAT_AccountBrokerStatus_LDTS] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = OFF, ALLOW_PAGE_LOCKS = OFF,
    DATA_COMPRESSION = PAGE) ON [PRIMARY]
    ) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

```

```

/***** Object: View [dbo].[V_HUB_Exchange]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

CREATE VIEW [dbo].[V_HUB_Exchange]
AS
SELECT
    ex.EX_ID ExchangeID
FROM
    TPCE.dbo.Exchange ex WITH (NOLOCK);
GO

```

```

/***** Object: View [dbo].[V_HUB_Customer]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

CREATE VIEW [dbo].[V_HUB_Customer]
AS
SELECT

```

```
        c.C_ID CustomerID
FROM
    TPCE.dbo.CUSTOMER c WITH (NOLOCK);
GO
```

/\*\*\*\*\*\* Object: View [dbo].[V\_SAT\_AccountCustomerStatus] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE VIEW [dbo].[V_SAT_AccountCustomerStatus]
AS
SELECT
    lnk.LNK_AccountCustomer_SQN,
    ca.CA_ID,
    ca.CA_C_ID
FROM
    TPCE.dbo.CUSTOMER_ACCOUNT ca WITH (NOLOCK)
    JOIN dbo.HUB_Account ha WITH (NOLOCK)
        ON ha.AccountID = ca.CA_ID
    JOIN dbo.HUB_Customer hc WITH (NOLOCK)
        ON hc.CustomerID = ca.CA_C_ID
    JOIN dbo.LNK_AccountCustomer lnk WITH (NOLOCK)
        ON lnk.HUB_Account_SQN = ha.HUB_Account_SQN
        AND lnk.HUB_Customer_SQN = hc.HUB_Customer_SQN;
GO
```

/\*\*\*\*\*\* Object: View [dbo].[V\_SAT\_ExchangeSecurities] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE VIEW [dbo].[V_SAT_ExchangeSecurities]
AS
SELECT
    hex.HUB_Exchange_SQN AS HUB_Exchange_SQN,
```

```
        ex.EX_NUM_SYMB AS ListedSecurities
FROM
    TPCE.dbo.EXCHANGE ex WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)
        ON hex.ExchangeID = ex.EX_ID;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_Exchange] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_SAT\_Exchange]

AS

SELECT

```
    hex.HUB_Exchange_SQN AS HUB_Exchange_SQN,
    ex.EX_NAME AS ExchangeName,
    ex.EX_DESC AS ExchangeDesc,
    ex.EX_OPEN AS OpeningTime,
    ex.EX_CLOSE AS ClosingTime,
    ad.AD_LINE1 AS AddressLine1,
    ad.AD_LINE2 AS AddressLine2,
    ad.AD_ZC_CODE AS PostalCode
```

FROM

```
    TPCE.dbo.EXCHANGE ex WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)
        ON hex.ExchangeID = ex.EX_ID
    JOIN TPCE.dbo.[ADDRESS] ad WITH (NOLOCK)
        ON ad.AD_ID = ex.EX_AD_ID;
```

GO

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_CustomerContact] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

```

CREATE VIEW [dbo].[V_SAT_CustomerContact]
AS
SELECT
    hc.HUB_Customer_SQN AS HUB_Customer_SQN,
    c.C_CTRY_1 AS PhoneCountryCodePrimary,
    c.C_AREA_1 AS PhoneAreaCodePrimary,
    c.C_LOCAL_1 AS PhoneLocalNumberPrimary,
    c.C_EXT_1 AS PhoneExtensionPrimary,
    c.C_CTRY_2 AS PhoneCountryCodeSecondary,
    c.C_AREA_2 AS PhoneAreaCodeSecondary,
    c.C_LOCAL_2 AS PhoneLocalNumberSecondary,
    c.C_EXT_2 AS PhoneExtensionSecondary,
    c.C_EMAIL_1 AS EmailPrimary,
    c.C_EMAIL_2 AS EmailSecondary,
    ad.AD_LINE1 AS AddressLine1,
    ad.AD_LINE2 AS AddressLine2,
    ad.AD_ZC_CODE AS PostalCode
FROM
    TPCE.dbo.CUSTOMER c WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Customer hc WITH (NOLOCK)
        ON hc.CustomerID = c.C_ID
    JOIN TPCE.dbo.[ADDRESS] ad WITH (NOLOCK)
        ON ad.AD_ID = c.C_AD_ID;
GO

```

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_Customer] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

```

SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

CREATE VIEW [dbo].[V_SAT_Customer]
AS
SELECT
    hc.HUB_Customer_SQN AS HUB_Customer_SQN,
    c.C_L_NAME AS LastName,
    c.C_F_NAME AS FirstName,
    c.C_M_NAME AS MiddleInitial,
    c.C_GNDR AS GENDER,

```

```

        c.C_DOB AS DateOfBirth,
        c.C_TIER AS TierNumber,
        c.C_ST_ID AS StatusCode
FROM
    TPCE.dbo.CUSTOMER c WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Customer hc WITH (NOLOCK)
        ON hc.CustomerID = c.C_ID;

```

GO

/\*\*\*\*\*\* Object: View [dbo].[V\_SAT\_SecurityCompanyStatus]      Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

```

SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO

```

```

CREATE VIEW [dbo].[V_SAT_SecurityCompanyStatus]
AS
SELECT
    lnk.LNK_SecurityCompany_SQN,
    hs.SecurityCode,
    hco.CompanyName
FROM
    TPCE.dbo.[SECURITY] s WITH (NOLOCK)
    JOIN TPCE.dbo.COMPANY co WITH (NOLOCK)
        ON co.CO_ID = s.S_CO_ID
    JOIN dbo.HUB_Company hco WITH (NOLOCK)
        ON hco.CompanyName = co.CO_NAME
    JOIN dbo.HUB_Security hs WITH (NOLOCK)
        ON hs.SecurityCode = s.S_SYMB
    JOIN dbo.LNK_SecurityCompany lnk WITH (NOLOCK)
        ON lnk.HUB_Security_SQN = hs.HUB_Security_SQN
        AND lnk.HUB_Company_SQN = hco.HUB_Company_SQN;

```

GO

/\*\*\*\*\*\* Object: View [dbo].[V\_SAT\_AccountBrokerStatus]      Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

```

SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON

```



GO

CREATE VIEW [dbo].[V\_SAT\_AccountBrokerStatus]

AS

SELECT

lnk.LNK\_AccountBroker\_SQN,  
ca.CA\_ID,  
ca.CA\_B\_ID

FROM

TPCE.dbo.CUSTOMER\_ACCOUNT ca WITH (NOLOCK)  
JOIN dbo.HUB\_Account ha WITH (NOLOCK)  
ON ha.AccountID = ca.CA\_ID  
JOIN dbo.HUB\_Broker hb WITH (NOLOCK)  
ON hb.BrokerID = ca.CA\_B\_ID  
JOIN dbo.LNK\_AccountBroker lnk WITH (NOLOCK)  
ON lnk.HUB\_Account\_SQN = ha.HUB\_Account\_SQN  
AND lnk.HUB\_Broker\_SQN = hb.HUB\_Broker\_SQN;

GO

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_SecurityExchangeStatus] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_SAT\_SecurityExchangeStatus]

AS

SELECT

lnk.LNK\_SecurityExchange\_SQN,  
hs.SecurityCode,  
hex.ExchangeID

FROM

TPCE.dbo.[SECURITY] s WITH (NOLOCK)  
JOIN dbo.HUB\_Security hs WITH (NOLOCK)  
ON hs.SecurityCode = s.S\_SYMB  
JOIN dbo.HUB\_Exchange hex  
ON hex.ExchangeID = s.S\_EX\_ID  
JOIN dbo.LNK\_SecurityExchange lnk WITH (NOLOCK)  
ON lnk.HUB\_Security\_SQN = hs.HUB\_Security\_SQN

```
        AND lnk.HUB_Exchange_SQN = hex.HUB_Exchange_SQN;  
GO
```

```
/****** Object: View [dbo].[V_SAT_SecurityCompanyStatusDetail]    Script Date: 11/30/2011 10:57:22 *****/  
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
CREATE VIEW [dbo].[V_SAT_SecurityCompanyStatusDetail]  
AS  
SELECT
```

```
    sat.LNK_SecurityCompany_SQN,  
    hs.SecurityCode,  
    hco.CompanyName,  
    sat.SAT_SecurityCompanyStatus_LDTS,  
    sat.SAT_SecurityCompanyStatus_LEDTS,  
    sat.SAT_SecurityCompanyStatus_RSRC
```

```
FROM  
    dbo.SAT_SecurityCompanyStatus sat  
JOIN  dbo.LNK_SecurityCompany lnk  
    ON lnk.LNK_SecurityCompany_SQN = sat.LNK_SecurityCompany_SQN  
JOIN  dbo.HUB_Security hs  
    ON hs.HUB_Security_SQN = lnk.HUB_Security_SQN  
JOIN  dbo.HUB_Company hco  
    ON hco.HUB_Company_SQN = lnk.HUB_Company_SQN
```

```
GO
```

```
/****** Object: View [dbo].[V_SAT_SecurityExchangeStatusDetail]  Script Date: 11/30/2011 10:57:22 *****/  
SET ANSI_NULLS ON  
GO
```

```
SET QUOTED_IDENTIFIER ON  
GO
```

```
CREATE VIEW [dbo].[V_SAT_SecurityExchangeStatusDetail]  
AS  
SELECT
```

```
    sat.LNK_SecurityExchange_SQN,
```

```

        hs.SecurityCode,
        hex.ExchangeID,
        sat.SAT_SecurityExchangeStatus_LDTS,
        sat.SAT_SecurityExchangeStatus_LEDTS,
        sat.SAT_SecurityExchangeStatus_RSRC
FROM
    dbo.SAT_SecurityExchangeStatus sat
JOIN  dbo.LNK_SecurityExchange lnk
        ON lnk.LNK_SecurityExchange_SQN = sat.LNK_SecurityExchange_SQN
JOIN  dbo.HUB_Security hs
        ON hs.HUB_Security_SQN = lnk.HUB_Security_SQN
JOIN  dbo.HUB_Exchange hex
        ON hex.HUB_Exchange_SQN = lnk.HUB_Exchange_SQN
GO

/***** Object: View [dbo].[V_SAT_AccountCustomerStatusDetail]    Script Date: 11/30/2011 10:57:22 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[V_SAT_AccountCustomerStatusDetail]
AS
SELECT
    sat.LNK_AccountCustomer_SQN,
    ha.AccountID,
    hc.CustomerID,
    sat.SAT_AccountCustomerStatus_LDTS,
    sat.SAT_AccountCustomerStatus_LEDTS,
    sat.SAT_AccountCustomerStatus_RSRC
FROM
    dbo.SAT_AccountCustomerStatus sat
JOIN  dbo.LNK_AccountCustomer lnk
        ON lnk.LNK_AccountCustomer_SQN = sat.LNK_AccountCustomer_SQN
JOIN  dbo.HUB_Account ha
        ON ha.HUB_Account_SQN = lnk.HUB_Account_SQN
JOIN  dbo.HUB_Customer hc
        ON hc.HUB_Customer_SQN = lnk.HUB_Customer_SQN
GO

```

/\*\*\*\*\* Object: View [dbo].[V\_SAT\_AccountBrokerStatusDetail] Script Date: 11/30/2011 10:57:22 \*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[V\_SAT\_AccountBrokerStatusDetail]

AS

SELECT

sat.LNK\_AccountBroker\_SQN,  
ha.AccountID,  
hb.BrokerID,  
sat.SAT\_AccountBrokerStatus\_LDTS,  
sat.SAT\_AccountBrokerStatus\_LEDTS,  
sat.SAT\_AccountBrokerStatus\_RSRC

FROM

dbo.SAT\_AccountBrokerStatus sat  
JOIN dbo.LNK\_AccountBroker lnk  
ON lnk.LNK\_AccountBroker\_SQN = sat.LNK\_AccountBroker\_SQN  
JOIN dbo.HUB\_Account ha  
ON ha.HUB\_Account\_SQN = lnk.HUB\_Account\_SQN  
JOIN dbo.HUB\_Broker hb  
ON hb.HUB\_Broker\_SQN = lnk.HUB\_Broker\_SQN

GO

/\*\*\*\*\* Object: ForeignKey [FK\_SAT\_Account\_HUB\_Account] Script Date: 11/30/2011 10:57:21 \*\*\*\*\*/

ALTER TABLE [dbo].[SAT\_Account] WITH NOCHECK ADD CONSTRAINT [FK\_SAT\_Account\_HUB\_Account] FOREIGN KEY([HUB\_Account\_SQN])

REFERENCES [dbo].[HUB\_Account] ([HUB\_Account\_SQN])

NOT FOR REPLICATION

GO

ALTER TABLE [dbo].[SAT\_Account] NOCHECK CONSTRAINT [FK\_SAT\_Account\_HUB\_Account]

GO

/\*\*\*\*\* Object: ForeignKey [FK\_LNK\_SecurityExchange\_HUB\_Exchange] Script Date: 11/30/2011 10:57:21 \*\*\*\*\*/

ALTER TABLE [dbo].[LNK\_SecurityExchange] WITH NOCHECK ADD CONSTRAINT [FK\_LNK\_SecurityExchange\_HUB\_Exchange] FOREIGN

KEY([HUB\_Exchange\_SQN])

REFERENCES [dbo].[HUB\_Exchange] ([HUB\_Exchange\_SQN])

```
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[LNK_SecurityExchange] NOCHECK CONSTRAINT [FK_LNK_SecurityExchange_HUB_Exchange]
GO
```

```
/****** Object: ForeignKey [FK_LNK_SecurityExchange_HUB_Security]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[LNK_SecurityExchange] WITH NOCHECK ADD CONSTRAINT [FK_LNK_SecurityExchange_HUB_Security] FOREIGN
KEY([HUB_Security_SQN])
REFERENCES [dbo].[HUB_Security] ([HUB_Security_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[LNK_SecurityExchange] NOCHECK CONSTRAINT [FK_LNK_SecurityExchange_HUB_Security]
GO
```

```
/****** Object: ForeignKey [FK_LNK_SecurityCompany_HUB_Company]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[LNK_SecurityCompany] WITH NOCHECK ADD CONSTRAINT [FK_LNK_SecurityCompany_HUB_Company] FOREIGN
KEY([HUB_Company_SQN])
REFERENCES [dbo].[HUB_Company] ([HUB_Company_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[LNK_SecurityCompany] NOCHECK CONSTRAINT [FK_LNK_SecurityCompany_HUB_Company]
GO
```

```
/****** Object: ForeignKey [FK_LNK_SecurityCompany_HUB_Security]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[LNK_SecurityCompany] WITH NOCHECK ADD CONSTRAINT [FK_LNK_SecurityCompany_HUB_Security] FOREIGN
KEY([HUB_Security_SQN])
REFERENCES [dbo].[HUB_Security] ([HUB_Security_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[LNK_SecurityCompany] NOCHECK CONSTRAINT [FK_LNK_SecurityCompany_HUB_Security]
GO
```

```
/****** Object: ForeignKey [FK_LNK_AccountCustomer_HUB_Account]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[LNK_AccountCustomer] WITH NOCHECK ADD CONSTRAINT [FK_LNK_AccountCustomer_HUB_Account] FOREIGN
KEY([HUB_Account_SQN])
REFERENCES [dbo].[HUB_Account] ([HUB_Account_SQN])
```

```
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[LNK_AccountCustomer] NOCHECK CONSTRAINT [FK_LNK_AccountCustomer_HUB_Account]
GO
```

```
/****** Object: ForeignKey [FK_LNK_AccountCustomer_HUB_Customer]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[LNK_AccountCustomer] WITH NOCHECK ADD CONSTRAINT [FK_LNK_AccountCustomer_HUB_Customer] FOREIGN
KEY([HUB_Customer_SQN])
REFERENCES [dbo].[HUB_Customer] ([HUB_Customer_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[LNK_AccountCustomer] NOCHECK CONSTRAINT [FK_LNK_AccountCustomer_HUB_Customer]
GO
```

```
/****** Object: ForeignKey [FK_LNK_AccountBroker_HUB_Account]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[LNK_AccountBroker] WITH NOCHECK ADD CONSTRAINT [FK_LNK_AccountBroker_HUB_Account] FOREIGN KEY([HUB_Account_SQN])
REFERENCES [dbo].[HUB_Account] ([HUB_Account_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[LNK_AccountBroker] NOCHECK CONSTRAINT [FK_LNK_AccountBroker_HUB_Account]
GO
```

```
/****** Object: ForeignKey [FK_LNK_AccountBroker_HUB_Broker]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[LNK_AccountBroker] WITH NOCHECK ADD CONSTRAINT [FK_LNK_AccountBroker_HUB_Broker] FOREIGN KEY([HUB_Broker_SQN])
REFERENCES [dbo].[HUB_Broker] ([HUB_Broker_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[LNK_AccountBroker] NOCHECK CONSTRAINT [FK_LNK_AccountBroker_HUB_Broker]
GO
```

```
/****** Object: ForeignKey [FK_TLNK_Trade_HUB_Account]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[TLNK_Trade] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_Trade_HUB_Account] FOREIGN KEY([HUB_Account_SQN])
REFERENCES [dbo].[HUB_Account] ([HUB_Account_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_Trade] NOCHECK CONSTRAINT [FK_TLNK_Trade_HUB_Account]
GO
```

```
/****** Object: ForeignKey [FK_TLNK_Trade_HUB_Broker]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[TLNK_Trade] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_Trade_HUB_Broker] FOREIGN KEY([HUB_Broker_SQN])
REFERENCES [dbo].[HUB_Broker] ([HUB_Broker_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_Trade] NOCHECK CONSTRAINT [FK_TLNK_Trade_HUB_Broker]
GO
```

```
/****** Object: ForeignKey [FK_TLNK_Trade_HUB_Company]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[TLNK_Trade] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_Trade_HUB_Company] FOREIGN KEY([HUB_Company_SQN])
REFERENCES [dbo].[HUB_Company] ([HUB_Company_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_Trade] NOCHECK CONSTRAINT [FK_TLNK_Trade_HUB_Company]
GO
```

```
/****** Object: ForeignKey [FK_TLNK_Trade_HUB_Customer]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[TLNK_Trade] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_Trade_HUB_Customer] FOREIGN KEY([HUB_Customer_SQN])
REFERENCES [dbo].[HUB_Customer] ([HUB_Customer_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_Trade] NOCHECK CONSTRAINT [FK_TLNK_Trade_HUB_Customer]
GO
```

```
/****** Object: ForeignKey [FK_TLNK_Trade_HUB_Exchange]    Script Date: 11/30/2011 10:57:21 *****/
ALTER TABLE [dbo].[TLNK_Trade] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_Trade_HUB_Exchange] FOREIGN KEY([HUB_Exchange_SQN])
REFERENCES [dbo].[HUB_Exchange] ([HUB_Exchange_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_Trade] NOCHECK CONSTRAINT [FK_TLNK_Trade_HUB_Exchange]
GO
```

```
/****** Object: ForeignKey [FK_TLNK_Trade_HUB_Security]    Script Date: 11/30/2011 10:57:21 *****/
```

```
ALTER TABLE [dbo].[TLNK_Trade] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_Trade_HUB_Security] FOREIGN KEY([HUB_Security_SQN])
REFERENCES [dbo].[HUB_Security] ([HUB_Security_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_Trade] NOCHECK CONSTRAINT [FK_TLNK_Trade_HUB_Security]
GO
```

```
/***** Object: ForeignKey [FK_TLNK_Trade_HUB_TradeType] Script Date: 11/30/2011 10:57:21 */
ALTER TABLE [dbo].[TLNK_Trade] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_Trade_HUB_TradeType] FOREIGN KEY([HUB_TradeType_SQN])
REFERENCES [dbo].[HUB_TradeType] ([HUB_TradeType_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_Trade] NOCHECK CONSTRAINT [FK_TLNK_Trade_HUB_TradeType]
GO
```

```
/***** Object: ForeignKey [FK_TLNK_CompanyPerformance_HUB_Company] Script Date: 11/30/2011 10:57:21 */
ALTER TABLE [dbo].[TLNK_QuarterlyCompanyPerformance] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_CompanyPerformance_HUB_Company] FOREIGN
KEY([HUB_Company_SQN])
REFERENCES [dbo].[HUB_Company] ([HUB_Company_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_QuarterlyCompanyPerformance] NOCHECK CONSTRAINT [FK_TLNK_CompanyPerformance_HUB_Company]
GO
```

```
/***** Object: ForeignKey [FK_TLNK_MarketPerformance_HUB_Company] Script Date: 11/30/2011 10:57:22 */
ALTER TABLE [dbo].[TLNK_DailyMarketPerformance] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_MarketPerformance_HUB_Company] FOREIGN
KEY([HUB_Company_SQN])
REFERENCES [dbo].[HUB_Company] ([HUB_Company_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_DailyMarketPerformance] NOCHECK CONSTRAINT [FK_TLNK_MarketPerformance_HUB_Company]
GO
```

```
/***** Object: ForeignKey [FK_TLNK_MarketPerformance_HUB_Exchange] Script Date: 11/30/2011 10:57:22 */
ALTER TABLE [dbo].[TLNK_DailyMarketPerformance] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_MarketPerformance_HUB_Exchange] FOREIGN
KEY([HUB_Exchange_SQN])
```



```
REFERENCES [dbo].[HUB_Exchange] ([HUB_Exchange_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_DailyMarketPerformance] NOCHECK CONSTRAINT [FK_TLNK_MarketPerformance_HUB_Exchange]
GO
```

```
/****** Object: ForeignKey [FK_TLNK_MarketPerformance_HUB_Security]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[TLNK_DailyMarketPerformance] WITH NOCHECK ADD CONSTRAINT [FK_TLNK_MarketPerformance_HUB_Security] FOREIGN
KEY([HUB_Security_SQN])
REFERENCES [dbo].[HUB_Security] ([HUB_Security_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[TLNK_DailyMarketPerformance] NOCHECK CONSTRAINT [FK_TLNK_MarketPerformance_HUB_Security]
GO
```

```
/****** Object: ForeignKey [FK_SAT_Security_HUB_Security]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_Security] WITH NOCHECK ADD CONSTRAINT [FK_SAT_Security_HUB_Security] FOREIGN KEY([HUB_Security_SQN])
REFERENCES [dbo].[HUB_Security] ([HUB_Security_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_Security] NOCHECK CONSTRAINT [FK_SAT_Security_HUB_Security]
GO
```

```
/****** Object: ForeignKey [FK_SAT_ExchangeSecurities_HUB_Exchange]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_ExchangeSecurities] WITH NOCHECK ADD CONSTRAINT [FK_SAT_ExchangeSecurities_HUB_Exchange] FOREIGN
KEY([HUB_Exchange_SQN])
REFERENCES [dbo].[HUB_Exchange] ([HUB_Exchange_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_ExchangeSecurities] NOCHECK CONSTRAINT [FK_SAT_ExchangeSecurities_HUB_Exchange]
GO
```

```
/****** Object: ForeignKey [FK_SAT_Exchange_HUB_Exchange]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_Exchange] WITH NOCHECK ADD CONSTRAINT [FK_SAT_Exchange_HUB_Exchange] FOREIGN KEY([HUB_Exchange_SQN])
REFERENCES [dbo].[HUB_Exchange] ([HUB_Exchange_SQN])
NOT FOR REPLICATION
```

GO

```
ALTER TABLE [dbo].[SAT_Exchange] NOCHECK CONSTRAINT [FK_SAT_Exchange_HUB_Exchange]
GO
```

```
/****** Object: ForeignKey [FK_SAT_CustomerContact_HUB_Customer]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_CustomerContact] WITH NOCHECK ADD CONSTRAINT [FK_SAT_CustomerContact_HUB_Customer] FOREIGN
KEY([HUB_Customer_SQN])
REFERENCES [dbo].[HUB_Customer] ([HUB_Customer_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_CustomerContact] NOCHECK CONSTRAINT [FK_SAT_CustomerContact_HUB_Customer]
GO
```

```
/****** Object: ForeignKey [FK_SAT_Customer_HUB_Customer]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_Customer] WITH NOCHECK ADD CONSTRAINT [FK_SAT_Customer_HUB_Customer] FOREIGN KEY([HUB_Customer_SQN])
REFERENCES [dbo].[HUB_Customer] ([HUB_Customer_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_Customer] NOCHECK CONSTRAINT [FK_SAT_Customer_HUB_Customer]
GO
```

```
/****** Object: ForeignKey [FK_SAT_CompanyCreditRating_HUB_Company]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_CompanyCreditRating] WITH NOCHECK ADD CONSTRAINT [FK_SAT_CompanyCreditRating_HUB_Company] FOREIGN
KEY([HUB_Company_SQN])
REFERENCES [dbo].[HUB_Company] ([HUB_Company_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_CompanyCreditRating] NOCHECK CONSTRAINT [FK_SAT_CompanyCreditRating_HUB_Company]
GO
```

```
/****** Object: ForeignKey [FK_SAT_CompanyCEO_HUB_Company]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_CompanyCEO] WITH NOCHECK ADD CONSTRAINT [FK_SAT_CompanyCEO_HUB_Company] FOREIGN KEY([HUB_Company_SQN])
REFERENCES [dbo].[HUB_Company] ([HUB_Company_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_CompanyCEO] NOCHECK CONSTRAINT [FK_SAT_CompanyCEO_HUB_Company]
GO
```

```
/****** Object: ForeignKey [FK_SAT_Company_HUB_Company]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_Company] WITH NOCHECK ADD CONSTRAINT [FK_SAT_Company_HUB_Company] FOREIGN KEY([HUB_Company_SQN])
REFERENCES [dbo].[HUB_Company] ([HUB_Company_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_Company] NOCHECK CONSTRAINT [FK_SAT_Company_HUB_Company]
GO
```

```
/****** Object: ForeignKey [FK_SAT_Broker_HUB_Broker]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_Broker] WITH NOCHECK ADD CONSTRAINT [FK_SAT_Broker_HUB_Broker] FOREIGN KEY([HUB_Broker_SQN])
REFERENCES [dbo].[HUB_Broker] ([HUB_Broker_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_Broker] NOCHECK CONSTRAINT [FK_SAT_Broker_HUB_Broker]
GO
```

```
/****** Object: ForeignKey [FK_SAT_TradeType_HUB_TradeType]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_TradeType] WITH NOCHECK ADD CONSTRAINT [FK_SAT_TradeType_HUB_TradeType] FOREIGN KEY([HUB_TradeType_SQN])
REFERENCES [dbo].[HUB_TradeType] ([HUB_TradeType_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_TradeType] NOCHECK CONSTRAINT [FK_SAT_TradeType_HUB_TradeType]
GO
```

```
/****** Object: ForeignKey [FK_SAT_SecurityExchangeStatus_LNK_SecurityExchange]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_SecurityExchangeStatus] WITH NOCHECK ADD CONSTRAINT [FK_SAT_SecurityExchangeStatus_LNK_SecurityExchange]
FOREIGN KEY([LNK_SecurityExchange_SQN])
REFERENCES [dbo].[LNK_SecurityExchange] ([LNK_SecurityExchange_SQN])
NOT FOR REPLICATION
GO
```

```
ALTER TABLE [dbo].[SAT_SecurityExchangeStatus] NOCHECK CONSTRAINT [FK_SAT_SecurityExchangeStatus_LNK_SecurityExchange]
GO
```

```

/***** Object: ForeignKey [FK_SAT_SecurityCompanyStatus_LNK_SecurityCompany]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_SecurityCompanyStatus] WITH NOCHECK ADD CONSTRAINT [FK_SAT_SecurityCompanyStatus_LNK_SecurityCompany]
FOREIGN KEY([LNK_SecurityCompany_SQN])
REFERENCES [dbo].[LNK_SecurityCompany] ([LNK_SecurityCompany_SQN])
NOT FOR REPLICATION
GO

ALTER TABLE [dbo].[SAT_SecurityCompanyStatus] NOCHECK CONSTRAINT [FK_SAT_SecurityCompanyStatus_LNK_SecurityCompany]
GO

/***** Object: ForeignKey [FK_SAT_AccountCustomerStatus_LNK_AccountCustomer]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_AccountCustomerStatus] WITH NOCHECK ADD CONSTRAINT [FK_SAT_AccountCustomerStatus_LNK_AccountCustomer]
FOREIGN KEY([LNK_AccountCustomer_SQN])
REFERENCES [dbo].[LNK_AccountCustomer] ([LNK_AccountCustomer_SQN])
NOT FOR REPLICATION
GO

ALTER TABLE [dbo].[SAT_AccountCustomerStatus] NOCHECK CONSTRAINT [FK_SAT_AccountCustomerStatus_LNK_AccountCustomer]
GO

/***** Object: ForeignKey [FK_SAT_AccountBrokerStatus_LNK_AccountBroker]    Script Date: 11/30/2011 10:57:22 *****/
ALTER TABLE [dbo].[SAT_AccountBrokerStatus] WITH NOCHECK ADD CONSTRAINT [FK_SAT_AccountBrokerStatus_LNK_AccountBroker] FOREIGN
KEY([LNK_AccountBroker_SQN])
REFERENCES [dbo].[LNK_AccountBroker] ([LNK_AccountBroker_SQN])
NOT FOR REPLICATION
GO

ALTER TABLE [dbo].[SAT_AccountBrokerStatus] NOCHECK CONSTRAINT [FK_SAT_AccountBrokerStatus_LNK_AccountBroker]
GO

```

## 1.2 Data Loading Scripts

## 1.2.1 Historical Data Loading Scripts

### 1.2.1.1 TPCE Historical Data Import T-SQL Script

```
USE TPCE;  
GO
```

```
BULK INSERT dbo.[ADDRESS] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Address.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[BROKER] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Broker.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[COMPANY] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Company.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[CUSTOMER] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Customer.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[CUSTOMER_ACCOUNT] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Customer_Account.txt' WITH  
(FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[DAILY_MARKET] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Daily_Market.txt' WITH (FIELDTERMINATOR =  
'|', ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[EXCHANGE] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Exchange.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[FINANCIAL] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Financial.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[INDUSTRY] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Industry.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[SECTOR] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Sector.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[SECURITY] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Security.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[STATUS_TYP] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Status_Type.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[TRADE] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Trade.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[TRADE_TYPE] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Trade_Type.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

```
BULK INSERT dbo.[ZIP_CODE] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Zip_Code.txt' WITH (FIELDTERMINATOR = '|',  
ROWTERMINATOR = '\n')
```

### 1.2.1.2 TPCE\_DM Historical Data Loading T-SQL Script

```
/*  
D_TradeType  
*/  
USE TPCE;  
GO  
  
USE TPCE;  
GO  
  
SET NOCOUNT ON;  
  
DECLARE  
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);  
  
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';  
  
INSERT INTO TPCE_DM.dbo.D_TradeType WITH (TABLOCK)  
(  
    TradeTypeCode,  
    TradeTypeDesc,
```

```

        TradeCategory,
        MarketTransactionFlag,
        WH_DateInserted
    )
SELECT
    tt.TT_ID AS TradeTypeCode,
    tt.TT_NAME AS TradeTypeDesc,
    CASE tt.TT_IS_SELL
        WHEN 0 THEN 'Buy'
    ELSE 'Sell'
    END AS TradeCategory,
    tt.TT_IS_MRKT AS MarketTransactionFlag,
    @BatchDate AS WH_DateInserted
FROM
    TPCE.dbo.TRADE_TYPE tt WITH (NOLOCK)

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_TradeType;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_TradeType',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

```

/*****
D_TradeStatus
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DM.dbo.D_TradeStatus WITH (TABLOCK)
(
    TradeStatusCode,
    TradeStatusDesc,
    WH_DateInserted
)
SELECT
    st.ST_ID AS TradeStatusCode,
    st.ST_NAME AS TradeStatusDesc,
    @BatchDate AS WH_DateInserted
FROM
    dbo.STATUS_TYPE st WITH (NOLOCK)

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_TradeStatus;

INSERT INTO master.dbo.LoadResults
SELECT

```



```
@Iteration,  
    'DM',  
    'D_TradeStatus',  
    'Dimension',  
    @QueryStart,  
    @QueryEnd,  
    NULL,  
    NULL,  
    @CountTotal;
```

GO

```
/*  
D_Security  
*/
```

```
USE TPCE;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DM.dbo.D_Security WITH (TABLOCK)
```

```
(  
    SecurityCode,  
    SecurityName,  
    IssueType,  
    FirstTradeDate,  
    SecurityStatusCode,  
    SecurityStatusDesc,  
    WH_DateInserted  
)  
SELECT
```

```

        s.S_SYMB AS SecurityCode,
        s.S_NAME AS SecurityName,
        s.S_ISSUE AS IssueType,
        s.S_START_DATE AS FirstTradeDate,
        st.ST_ID AS SecurityStatusCode,
        st.ST_NAME AS SecurityStatusDesc,
        @BatchDate AS WH_DateInserted
FROM
    dbo.[SECURITY] s WITH (NOLOCK)
    JOIN dbo.STATUS_TYPE st WITH (NOLOCK)
        ON s.S_ST_ID = st.ST_ID

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_Security;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_Security',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
D_Exchange
*****/
USE TPCE;

```

```
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
```

```
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DM.dbo.D_Exchange WITH (TABLOCK)
```

```
(  
    ExchangeID,  
    ExchangeName,  
    ExchangeDesc,  
    OpeningTime,  
    ClosingTime,  
    ListedSecurities,  
    AddressLine1,  
    AddressLine2,  
    TownName,  
    StateProvinceCode,  
    PostalCode,  
    CountryName,  
    WH_DateInserted  
)
```

```
SELECT
```

```
    e.EX_ID AS ExchangeID,  
    e.EX_NAME AS ExchangeName,  
    e.EX_DESC AS ExchangeDesc,  
    CASE LEN(e.EX_OPEN)  
        WHEN 3 THEN '0' + LEFT(CONVERT(CHAR(3),e.EX_OPEN),1) + ':' + RIGHT(CONVERT(CHAR(3),e.EX_OPEN),2)  
        WHEN 4 THEN LEFT(CONVERT(CHAR(4),e.EX_OPEN),2) + ':' + RIGHT(CONVERT(CHAR(4),e.EX_OPEN),2)  
    END AS OpeningTime,  
    CASE LEN(e.EX_CLOSE)  
        WHEN 3 THEN '0' + LEFT(CONVERT(CHAR(3),e.EX_CLOSE),1) + ':' + RIGHT(CONVERT(CHAR(3),e.EX_CLOSE),2)  
        WHEN 4 THEN LEFT(CONVERT(CHAR(4),e.EX_CLOSE),2) + ':' + RIGHT(CONVERT(CHAR(4),e.EX_CLOSE),2)  
    END AS ClosingTime,
```

```

e.EX_NUM_SYMB AS ListedSecurities,
a.AD_LINE1 AS AddressLine1,
ISNULL(a.AD_LINE2, '') AS AddressLine2,
z.ZC_TOWN AS TownName,
z.ZC_DIV AS StateProvinceCode,
z.ZC_CODE AS PostalCode,
CASE UPPER(a.AD_CTRY)
    WHEN 'USA' THEN UPPER(a.AD_CTRY)
    WHEN NULL THEN 'Unknown'
ELSE
    UPPER(LEFT(a.AD_CTRY,1)) + LOWER(RIGHT(a.AD_CTRY,LEN(a.AD_CTRY)-1))
END AS CountryName,
@BatchDate AS WH_DateInserted
FROM
    dbo.EXCHANGE e WITH (NOLOCK)
JOIN dbo.[ADDRESS] a WITH (NOLOCK)
    ON e.EX_AD_ID = a.AD_ID
JOIN dbo.ZIP_CODE z WITH (NOLOCK)
    ON a.AD_ZC_CODE = z.ZC_CODE

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_Exchange;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_Exchange',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

GO

```
/*  
D_Date  
*/
```

```
USE TPCE;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @StartDate DATE,  
    @EndDate DATE,  
    @CurrentDate DATE,  
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';  
SET @StartDate = '1999/01/01';  
SET @EndDate = '2010/01/01';  
SET @CurrentDate = '1999/01/01';
```

```
WHILE @CurrentDate < @EndDate  
BEGIN
```

```
    INSERT INTO TPCE_DM.dbo.D_Date WITH (TABLOCK)  
    (  
        Date_SK,  
        [Date],  
        DayOfWeekNumber,  
        DayOfWeekName,  
        DayOfMonthNumber,  
        DayOfYearNumber,  
        [MonthName],  
        MonthOfYearNumber,  
        [Quarter],
```

```

        [Year],
        WH_DateInserted,
        WH_DateUpdated
    )
VALUES
(
    CONVERT(INT, CONVERT(VARCHAR(10), @CurrentDate, 112)),
    @CurrentDate,
    DATEPART(WEEKDAY, @CurrentDate),
    DATENAME(WEEKDAY, @CurrentDate),
    DATEPART(DAY, @CurrentDate),
    DATEPART(DAYOFYEAR, @CurrentDate),
    DATENAME(MONTH, @CurrentDate),
    DATEPART(MONTH, @CurrentDate),
    DATEPART(QUARTER, @CurrentDate),
    DATEPART(YEAR, @CurrentDate),
    @BatchDate,
    NULL
);

SET @CurrentDate = DATEADD(DAY, 1, @CurrentDate);

END;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_Date;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_Date',
    'Dimension',
    @QueryStart,
    @QueryEnd,

```

```
NULL,  
NULL,  
@CountTotal;
```

```
GO
```

```
/*  
D_FiscalQuarter  
*/
```

```
USE TPCE;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DM.dbo.D_FiscalQuarter WITH (TABLOCK)  
SELECT DISTINCT
```

```
    (Year * 10) + Quarter AS Quarter_SK,  
    Quarter,  
    Year,  
    @BatchDate,  
    NULL
```

```
FROM
```

```
    TPCE_DM.dbo.D_Date WITH (NOLOCK)
```

```
ORDER BY  
    1;
```

```
SET @QueryEnd = SYSDATETIME();
```

```
DECLARE
```

```
    @Iteration AS INT,  
    @CountTotal AS INT;
```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_FiscalQuarter;
```

```
INSERT INTO master.dbo.LoadResults
SELECT
```

```
    @Iteration,
    'DM',
    'D_FiscalQuarter',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;
```

```
GO
```

```

/*****
D_CustomerAccount
*****/
```

```
USE TPCE;
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DM.dbo.D_CustomerAccount WITH (TABLOCK)
(
    AccountID,
    AccountName,
    TaxStatusCode,
```



```

TaxStatusDesc,
CustomerID,
LastName,
FirstName,
MiddleInitial,
GENDER,
DateOfBirth,
TierNumber,
TierDesc,
PhonePrimary,
PhoneSecondary,
EmailPrimary,
EmailSecondary,
AddressLine1,
AddressLine2,
TownName,
StateProvinceCode,
PostalCode,
CountryName,
CustomerStatusCode,
CustomerStatusDesc,
WH_StartDate,
WH_CurrInd

```

```
)
```

```
SELECT
```

```

a.CA_ID AS AccountID,
a.CA_NAME AS AccountName,
a.CA_TAX_ST AS TaxStatusCode,
CASE a.CA_TAX_ST
    WHEN 0 THEN 'Not taxable.'
    WHEN 1 THEN 'Taxable. Withhold tax.'
    WHEN 2 THEN 'Taxable. Do not withhold tax.'
ELSE
    'Unknown'
END AS TaxStatusDesc,
c.C_ID AS CustomerID,
c.C_L_NAME AS LastName,
c.C_F_NAME AS FirstName,
ISNULL(c.C_M_NAME, '') AS MiddleInitial,
CASE c.C_GNDR

```

```

        WHEN 'M' THEN 'Male'
        WHEN 'F' THEN 'Female'
        ELSE 'Unknown'
    END AS GENDER,
    c.C_DOB AS DateOfBirth,
    c.C_TIER AS TierNumber,
    CASE c.C_TIER
        WHEN 1 THEN 'Has between 1 and 4 accounts.'
        WHEN 2 THEN 'Has between 2 and 8 accounts.'
        WHEN 3 THEN 'Has between 5 and 10 accounts.'
        ELSE 'Unknown'
    END AS TierDesc,
    CASE
        WHEN c.C_CTRY_1 IS NULL AND c.C_AREA_1 IS NULL AND c.C_LOCAL_1 IS NULL
            THEN 'Unknown'
        WHEN c.C_EXT_1 IS NULL
            THEN c.C_CTRY_1 + '-' + c.C_AREA_1 + '-' + LEFT(c.C_LOCAL_1,3) + '-' + RIGHT(c.C_LOCAL_1,4)
        ELSE
            c.C_CTRY_1 + '-' + c.C_AREA_1 + '-' + LEFT(c.C_LOCAL_1,3) + '-' + RIGHT(c.C_LOCAL_1,4) + ' ext. ' + c.C_EXT_1
    END AS PhonePrimary,
    CASE
        WHEN c.C_CTRY_2 IS NULL AND c.C_AREA_2 IS NULL AND c.C_LOCAL_2 IS NULL
            THEN 'Unknown'
        WHEN c.C_EXT_2 IS NULL
            THEN c.C_CTRY_2 + '-' + c.C_AREA_2 + '-' + LEFT(c.C_LOCAL_2,3) + '-' + RIGHT(c.C_LOCAL_2,4)
        ELSE
            c.C_CTRY_2 + '-' + c.C_AREA_2 + '-' + LEFT(c.C_LOCAL_2,3) + '-' + RIGHT(c.C_LOCAL_2,4) + ' ext. ' + c.C_EXT_2
    END AS PhoneSecondary,
    ISNULL(c.C_EMAIL_1, 'Unknown') AS EmailPrimary,
    ISNULL(c.C_EMAIL_2, 'Unknown') AS EmailSecondary,
    ad.AD_LINE1 AS AddressLine1,
    ISNULL(ad.AD_LINE2, '') AS AddressLine2,
    z.ZC_TOWN AS TownName,
    z.ZC_DIV AS StateProvinceCode,
    ad.AD_ZC_CODE AS PostalCode,
    CASE UPPER(ad.AD_CTRY)
        WHEN 'USA' THEN UPPER(ad.AD_CTRY)
        WHEN NULL THEN 'Unknown'
    ELSE
        UPPER(LEFT(ad.AD_CTRY,1)) + LOWER(RIGHT(ad.AD_CTRY, LEN(ad.AD_CTRY)-1))
    END

```

```

        END AS CountryName,
        st.ST_ID AS CustomerStatusCode,
        st.ST_NAME AS CustomerStatusDesc,
        @BatchDate AS WH_StartDate,
        1 AS WH_CurrInd
FROM
    dbo.CUSTOMER_ACCOUNT a WITH (NOLOCK)
    JOIN dbo.CUSTOMER c WITH (NOLOCK)
        ON c.C_ID = a.CA_C_ID
    JOIN dbo.[ADDRESS] ad WITH (NOLOCK)
        ON c.C_AD_ID = ad.AD_ID
    JOIN dbo.ZIP_CODE z WITH (NOLOCK)
        ON ad.AD_ZC_CODE = z.ZC_CODE
    JOIN dbo.STATUS_TYPE st WITH (NOLOCK)
        ON c.C_ST_ID = st.ST_ID

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_CustomerAccount;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_CustomerAccount',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

```

/*****
D_Company
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DM.dbo.D_Company WITH (TABLOCK)
(
    CompanyID,
    CompanyName,
    CompanyDesc,
    CompanyStatusCode,
    CompanyStatusDesc,
    CompanyCEO,
    CompanyCreditRating,
    CompanyOpenDate,
    IndustryCode,
    IndustryDesc,
    SectorCode,
    SectorDesc,
    AddressLine1,
    AddressLine2,
    TownName,
    StateProvinceCode,
    PostalCode,
    CountryName,
    WH_StartDate,
    WH_CurrInd
)
SELECT

```

```

c.CO_ID AS CompanyID,
c.CO_NAME AS CompanyName,
c.CO_DESC AS CompanyDesc,
st.ST_ID AS CompanyStatusCode,
st.ST_NAME AS CompanyStatusDesc,
c.CO_CEO AS CompanyCEO,
c.CO_SP_RATE AS CompanyCreditRating,
c.CO_OPEN_DATE AS CompanyOpenDate,
i.IN_ID AS IndustryCode,
i.IN_NAME AS IndustryDesc,
s.SC_ID AS SectorCode,
s.SC_NAME AS SectorDesc,
a.AD_LINE1 AS AddressLine1,
ISNULL(a.AD_LINE2, '') AS AddressLine2,
z.ZC_TOWN AS TownName,
z.ZC_DIV AS StateProvinceCode,
z.ZC_CODE AS PostalCode,
CASE UPPER(a.AD_CTRY)
    WHEN 'USA' THEN UPPER(a.AD_CTRY)
    WHEN NULL THEN 'Unknown'
ELSE
    UPPER(LEFT(a.AD_CTRY,1)) + LOWER(RIGHT(a.AD_CTRY,LEN(a.AD_CTRY)-1))
END AS CountryName,
@BatchDate AS WH_StartDate,
1 AS WH_CurrInd

```

FROM

```

dbo.COMPANY c WITH (NOLOCK)
JOIN dbo.INDUSTRY i WITH (NOLOCK)
    ON c.CO_IN_ID = i.IN_ID
JOIN dbo.SECTOR s WITH (NOLOCK)
    ON i.IN_SC_ID = s.SC_ID
JOIN dbo.STATUS_TYPE st WITH (NOLOCK)
    ON c.CO_ST_ID = st.ST_ID
JOIN dbo.[ADDRESS] a WITH (NOLOCK)
    ON c.CO_AD_ID = a.AD_ID
JOIN dbo.ZIP_CODE z WITH (NOLOCK)
    ON a.AD_ZC_CODE = z.ZC_CODE

```

```
SET @QueryEnd = SYSDATETIME();
```

```

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_Company;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_Company',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

```
GO
```

```

/*****
D_CashType
*****/

```

```

USE TPCE;
GO

```

```
SET NOCOUNT ON;
```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

```

```

INSERT INTO TPCE_DM.dbo.D_CashType WITH (TABLOCK)
(

```

```

        CashType,
        IsCashFlag,
        WH_DateInserted
    )
SELECT DISTINCT
    CASE t.T_IS_CASH
        WHEN 1 THEN 'Cash Account'
        WHEN 0 THEN 'Margin'
    END AS CashType,
    t.T_IS_CASH AS IsCashFlag,
    @BatchDate AS WH_DateInserted
FROM
    dbo.TRADE t WITH (NOLOCK)

```

```

SET @QueryEnd = SYSDATETIME();

```

```

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_CashType;

```

```

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_CashType',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

```

GO

```

```

/*****
D_Broker

```

```

*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DM.dbo.D_Broker WITH (TABLOCK)
(
    BrokerID,
    BrokerName,
    BrokerStatusCode,
    BrokerStatusDesc,
    WH_DateInserted
)
SELECT
    b.B_ID AS BrokerID,
    b.B_NAME AS BrokerName,
    s.ST_ID AS BrokerStatusCode,
    s.ST_NAME AS BrokerStatusDesc,
    @BatchDate AS WH_DateInserted
FROM
    dbo.[BROKER] b WITH (NOLOCK)
    JOIN dbo.STATUS_TYPE s WITH (NOLOCK)
        ON s.ST_ID = b.B_ST_ID

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;

```



```
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.D_Broker;
```

```
INSERT INTO master.dbo.LoadResults
```

```
SELECT
```

```
    @Iteration,  
    'DM',  
    'D_Broker',  
    'Dimension',  
    @QueryStart,  
    @QueryEnd,  
    NULL,  
    NULL,  
    @CountTotal;
```

```
GO
```

```
/*  
F_QuarterlyCompanyPerformance  
*/
```

```
USE TPCE;
```

```
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
```

```
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
ALTER INDEX IDX_F_QuarterlyCompanyPerformance_FiscalQuarter_SK ON TPCE_DM.dbo.F_QuarterlyCompanyPerformance DISABLE;
```

```
ALTER INDEX IDX_F_QuarterlyCompanyPerformance_Company_SK ON TPCE_DM.dbo.F_QuarterlyCompanyPerformance DISABLE;
```

```
ALTER INDEX IDX_F_QuarterlyCompanyPerformance_FiscalQuarterStartDate_SK ON TPCE_DM.dbo.F_QuarterlyCompanyPerformance DISABLE;
```

```
INSERT INTO TPCE_DM.dbo.F_QuarterlyCompanyPerformance WITH (TABLOCK)
```

```
(  
    FiscalQuarter_SK,
```

```

    Company_SK,
    FiscalQuarterStartDate_SK,
    Revenue,
    NetEarnings,
    EarningsPerShareBasic,
    EarningsPerShareDiluted,
    Margin,
    Inventory,
    Assets,
    Liabilities,
    SharesOutstandingBasicAvg,
    SharesOutstandingDilutedAvg,
    WH_DateInserted
)
SELECT
    (f.FI_YEAR * 10) + f.FI_QTR AS FiscalQuarter_SK,
    dc.Company_SK AS Company_SK,
    CONVERT(INT, CONVERT(VARCHAR(8), f.FI_QTR_START_DATE, 112)) AS QuarterStartDate_SK,
    f.FI_REVENUE AS Revenue,
    f.FI_NET_EARN AS NetEarnings,
    f.FI_BASIC_EPS AS EarningsPerShareBasic,
    f.FI_DILUT_EPS AS EarningsPerShareDiluted,
    f.FI_MARGIN AS Margin,
    f.FI_INVENTORY AS Inventory,
    f.FI_ASSETS AS Assets,
    f.FI_LIABILITY AS Liabilities,
    f.FI_OUT_BASIC AS SharesOutstandingBasicAvg,
    f.FI_OUT_DILUT AS SharesOutstandingDilutedAvg,
    @BatchDate AS WH_DateInserted
FROM
    dbo.FINANCIAL f WITH (NOLOCK)
    JOIN TPCE_DM.dbo.D_Company dc WITH (NOLOCK)
        ON f.FI_CO_ID = dc.CompanyID;

ALTER INDEX IDX_F_QuarterlyCompanyPerformance_FiscalQuarter_SK ON TPCE_DM.dbo.F_QuarterlyCompanyPerformance REBUILD;
ALTER INDEX IDX_F_QuarterlyCompanyPerformance_Company_SK ON TPCE_DM.dbo.F_QuarterlyCompanyPerformance REBUILD;
ALTER INDEX IDX_F_QuarterlyCompanyPerformance_FiscalQuarterStartDate_SK ON TPCE_DM.dbo.F_QuarterlyCompanyPerformance REBUILD;

SET @QueryEnd = SYSDATETIME();

```

```

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.F_QuarterlyCompanyPerformance;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'F_QuarterCompanyPerformance',
    'Fact',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
F_DailyMarketPerformance
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

ALTER INDEX IDX_F_DailyMarketPerformance_Date_SK ON TPCE_DM.dbo.F_DailyMarketPerformance DISABLE;
ALTER INDEX IDX_F_DailyMarketPerformance_Company_SK ON TPCE_DM.dbo.F_DailyMarketPerformance DISABLE;

```

```
ALTER INDEX IDX_F_DailyMarketPerformance_Security_SK ON TPCE_DM.dbo.F_DailyMarketPerformance DISABLE;
```

```
INSERT INTO TPCE_DM.dbo.F_DailyMarketPerformance WITH (TABLOCK)
```

```
(  
    Date_SK,  
    Company_SK,  
    Exchange_SK,  
    Security_SK,  
    DayClose,  
    DayHigh,  
    DayLow,  
    DayVolume,  
    [2WeekHigh],  
    [2WeekLow],  
    [2WeekVolume],  
    WH_DateInserted  
)
```

```
SELECT
```

```
    CONVERT(VARCHAR(8),dm.DM_DATE,112) AS Date_SK,  
    dc.Company_SK,  
    de.Exchange_SK,  
    ds.Security_SK,  
    dm.DM_CLOSE,  
    dm.DM_HIGH,  
    dm.DM_LOW,  
    dm.DM_VOL,  
    wk.[2WeekHigh],  
    wk.[2WeekLow],  
    wk.[2WeekVol],  
    @BatchDate
```

```
FROM
```

```
    dbo.DAILY_MARKET dm WITH (NOLOCK)  
  JOIN dbo.[SECURITY] s WITH (NOLOCK)  
        ON dm.DM_S_SYMB = s.S_SYMB  
  JOIN TPCE_DM.dbo.D_Security ds WITH (NOLOCK)  
        ON dm.DM_S_SYMB = ds.SecurityCode  
  JOIN TPCE_DM.dbo.D_Company dc WITH (NOLOCK)  
        ON s.S_CO_ID = dc.CompanyID  
  JOIN TPCE_DM.dbo.D_Exchange de WITH (NOLOCK)  
        ON s.S_EX_ID = de.ExchangeID
```

JOIN

(

SELECT

x.DM\_DATE,  
x.DM\_S\_SYMB,  
MIN(y.DM\_LOW) [2WeekLow],  
MAX(y.DM\_HIGH) [2WeekHigh],  
SUM(y.DM\_VOL) [2WeekVol]

FROM

dbo.DAILY\_MARKET x WITH (NOLOCK)  
JOIN dbo.DAILY\_MARKET y WITH (NOLOCK)  
ON x.DM\_S\_SYMB = y.DM\_S\_SYMB  
AND y.DM\_DATE >= DATEADD(WEEK,-2,x.DM\_DATE)  
AND y.DM\_DATE <= x.DM\_DATE

WHERE

x.DM\_DATE >= '2004/01/01'

GROUP BY

x.DM\_DATE,  
x.DM\_S\_SYMB

) wk

ON dm.DM\_DATE = wk.DM\_DATE

AND dm.DM\_S\_SYMB = wk.DM\_S\_SYMB;

ALTER INDEX IDX\_F\_DailyMarketPerformance\_Date\_SK ON TPCE\_DM.dbo.F\_DailyMarketPerformance REBUILD;

ALTER INDEX IDX\_F\_DailyMarketPerformance\_Company\_SK ON TPCE\_DM.dbo.F\_DailyMarketPerformance REBUILD;

ALTER INDEX IDX\_F\_DailyMarketPerformance\_Security\_SK ON TPCE\_DM.dbo.F\_DailyMarketPerformance REBUILD;

SET @QueryEnd = SYSDATETIME();

DECLARE

@Iteration AS INT,  
@CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;

SELECT @CountTotal = COUNT(\*) FROM TPCE\_DM.dbo.F\_DailyMarketPerformance;

INSERT INTO master.dbo.LoadResults

SELECT

@Iteration,  
'DM',

```
'F_DailyMarketPerformance',  
'Fact',  
@QueryStart,  
@QueryEnd,  
NULL,  
NULL,  
@CountTotal;
```

GO

```
/*  
F_Trade  
*/
```

```
USE TPCE;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
ALTER INDEX IDX_F_Trade_Date_SK ON TPCE_DM.dbo.F_Trade DISABLE;  
ALTER INDEX IDX_F_Trade_Broker_SK ON TPCE_DM.dbo.F_Trade DISABLE;  
ALTER INDEX IDX_F_Trade_Company_SK ON TPCE_DM.dbo.F_Trade DISABLE;  
ALTER INDEX IDX_F_Trade_CustomerAccount_SK ON TPCE_DM.dbo.F_Trade DISABLE;  
ALTER INDEX IDX_F_Trade_Security_SK ON TPCE_DM.dbo.F_Trade DISABLE;
```

```
INSERT INTO TPCE_DM.dbo.F_Trade WITH (TABLOCK)  
(
```

```
    Date_SK,  
    Broker_SK,  
    CustomerAccount_SK,  
    Company_SK,  
    Exchange_SK,
```

```
Security_SK,  
TradeType_SK,  
CashType_SK,  
TradeStatus_SK,  
TradeID,  
TradeDateTime,  
Quantity,  
BidPrice,  
TradePrice,  
Charge,  
Commission,  
Tax,  
WH_DateInserted
```

```
)
```

```
SELECT
```

```
    CONVERT(VARCHAR(8),t.T_DTS,112) AS Date_SK,  
    db.Broker_SK AS Broker_SK,  
    dca.CustomerAccount_SK AS CustomerAccount_SK,  
    dcom.Company_SK AS Company_SK,  
    de.Exchange_SK AS Exchange_SK,  
    ds.Security_SK AS Security_SK,  
    dtt.TradeType_SK AS TradeType_SK,  
    dct.CashType_SK AS CashType_SK,  
    dts.TradeStatus_SK AS TradeStatus_SK,  
    t.T_ID AS TradeID,  
    t.T_DTS AS TradeDateTime,  
    t.T_QTY AS Quantity,  
    t.T_BID_PRICE AS BidPrice,  
    t.T_TRADE_PRICE AS TradePrice,  
    t.T_CHRG AS Charge,  
    t.T_COMM AS Commission,  
    t.T_TAX AS Tax,  
    @BatchDate AS WH_DateInserted
```

```
FROM
```

```
    dbo.TRADE t (NOLOCK)  
    JOIN dbo.CUSTOMER_ACCOUNT a (NOLOCK)  
        ON t.T_CA_ID = a.CA_ID  
    JOIN TPCE_DM.dbo.D_CustomerAccount dca (NOLOCK)  
        ON a.CA_ID = dca.AccountID  
    JOIN dbo.[BROKER] b (NOLOCK)
```

```

        ON b.B_ID = a.CA_B_ID
JOIN TPCE_DM.dbo.D_Broker db (NOLOCK)
        ON db.BrokerID = b.B_ID
JOIN dbo.[SECURITY] s (NOLOCK)
        ON t.T_S_SYMB = s.S_SYMB
JOIN TPCE_DM.dbo.D_Company dcom (NOLOCK)
        ON s.S_CO_ID = dcom.CompanyID
JOIN TPCE_DM.dbo.D_Exchange de (NOLOCK)
        ON s.S_EX_ID = de.ExchangeID
JOIN TPCE_DM.dbo.D_Security ds (NOLOCK)
        ON s.S_SYMB = ds.SecurityCode
JOIN TPCE_DM.dbo.D_TradeType dtt (NOLOCK)
        ON t.T_TT_ID = dtt.TradeTypeCode
JOIN TPCE_DM.dbo.D_CashType dct (NOLOCK)
        ON t.T_IS_CASH = dct.IsCashFlag
JOIN dbo.STATUS_TYPE st (NOLOCK)
        ON t.T_ST_ID = st.ST_ID
JOIN TPCE_DM.dbo.D_TradeStatus dts (NOLOCK)
        ON st.ST_ID = dts.TradeStatusCode;

```

```

ALTER INDEX IDX_F_Trade_Date_SK ON TPCE_DM.dbo.F_Trade REBUILD;
ALTER INDEX IDX_F_Trade_Broker_SK ON TPCE_DM.dbo.F_Trade REBUILD;
ALTER INDEX IDX_F_Trade_Company_SK ON TPCE_DM.dbo.F_Trade REBUILD;
ALTER INDEX IDX_F_Trade_CustomerAccount_SK ON TPCE_DM.dbo.F_Trade REBUILD;
ALTER INDEX IDX_F_Trade_Security_SK ON TPCE_DM.dbo.F_Trade REBUILD;

```

```
SET @QueryEnd = SYSDATETIME();
```

```
DECLARE
```

```

    @Iteration AS INT,
    @CountTotal AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.F_Trade;

```

```

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'F_Trade',

```



```
'Fact',  
@QueryStart,  
@QueryEnd,  
NULL,  
NULL,  
@CountTotal;
```

```
GO
```

### 1.2.1.3 TPCE\_DV Historical Data Loading T-SQL Script

```
/*  
XREF_Status  
***/  
USE TPCE;  
GO  
  
USE TPCE;  
GO  
  
SET NOCOUNT ON;  
  
DECLARE  
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);  
  
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';  
  
INSERT INTO TPCE_DV.dbo.XREF_Status WITH (TABLOCK)  
(  
    StatusCode,  
    XREF_Status_LDTS,  
    XREF_Status_RSRC,  
    StatusDesc  
)  
SELECT
```

```

        st.ST_ID AS StatusCode,
        @BatchDate AS XREF_Status_LDTS,
        'TPCE' AS XREF_Status_RSRC,
        st.ST_NAME AS StatusDesc
FROM
    dbo.STATUS_TYPE st WITH (NOLOCK)

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.XREF_Status;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'XREF_Status',
    'XREF',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
XREF_Sector
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE

```

```

        @BatchDate AS DATETIME2(0),
        @QueryStart AS DATETIME2(7),
        @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.XREF_Sector WITH (TABLOCK)
(
    SectorCode,
    XREF_Sector_LDTS,
    XREF_Sector_RSRC,
    SectorDesc
)
SELECT
    sc.SC_ID AS SectorCode,
    @BatchDate AS XREF_Sector_LDTS,
    'TPCE' AS XREF_Sector_RSRC,
    sc.SC_NAME AS SectorDesc
FROM
    dbo.SECTOR sc

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.XREF_Sector;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'XREF_Sector',
    'XREF',
    @QueryStart,
    @QueryEnd,
    NULL,

```

```
NULL,  
@CountTotal;
```

```
GO
```

```
/*  
XREF_Industry  
***/
```

```
USE TPCE;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DV.dbo.XREF_Industry WITH (TABLOCK)
```

```
(  
    IndustryCode,  
    XREF_Industry_LDTS,  
    XREF_Industry_RSRC,  
    IndustryDesc  
)
```

```
SELECT  
    ind.IN_ID AS IndustryCode,  
    @BatchDate AS XREF_Industry_LDTS,  
    'TPCE' AS XREF_Industry_RSRC,  
    ind.IN_NAME AS IndustryDesc
```

```
FROM  
    dbo.INDUSTRY ind
```

```
SET @QueryEnd = SYSDATETIME();
```

```
DECLARE
```

```

        @Iteration AS INT,
        @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.XREF_Industry;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'XREF_Industry',
    'XREF',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
HUB_TradeType
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.HUB_TradeType WITH (TABLOCK)
(
    HUB_TradeType_LDTS,

```

```

        HUB_TradeType_RSRC,
        TradeTypeDesc
    )
SELECT
    @BatchDate AS HUB_TradeType_LDTS,
    'TPCE' AS HUB_TradeType_RSRC,
    tt.TT_NAME TradeTypeCode
FROM
    dbo.TRADE_TYPE tt WITH (NOLOCK)

```

```
SET @QueryEnd = SYSDATETIME();
```

```
DECLARE
```

```

    @Iteration AS INT,
    @CountTotal AS INT;

```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
```

```
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.HUB_TradeType;
```

```
INSERT INTO master.dbo.LoadResults
```

```

SELECT
    @Iteration,
    'DV',
    'HUB_TradeType',
    'HUB',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

```
GO
```

```

/*****
HUB_Security
*****/
USE TPCE;
GO

```

```

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.HUB_Security WITH (TABLOCK)
(
    HUB_Security_LDTS,
    HUB_Security_RSRC,
    SecurityCode
)
SELECT
    @BatchDate AS HUB_Security_LDTS,
    'TPCE' AS HUB_Security_RSRC,
    s.S_SYMB SecurityCode
FROM
    dbo.[SECURITY] s WITH (NOLOCK);

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.HUB_Security;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_Security',
    'HUB',

```

```
@QueryStart,  
@QueryEnd,  
NULL,  
NULL,  
@CountTotal;
```

```
GO
```

```
/*  
HUB_Exchange  
*/
```

```
USE TPCE;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DV.dbo.HUB_Exchange WITH (TABLOCK)
```

```
(  
    HUB_Exchange_LDTS,  
    HUB_Exchange_RSRC,  
    ExchangeID
```

```
)
```

```
SELECT
```

```
    @BatchDate AS HUB_Exchange_LDTS,  
    'TPCE' AS HUB_Exchange_RSRC,  
    ex.EX_ID ExchangeID
```

```
FROM
```

```
    dbo.Exchange ex WITH (NOLOCK)
```

```
SET @QueryEnd = SYSDATETIME();
```



```

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.HUB_Exchange;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_Exchange',
    'HUB',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

```
GO
```

```

/*****
HUB_Customer
*****/

```

```

USE TPCE;
GO

```

```
SET NOCOUNT ON;
```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.HUB_Customer WITH (TABLOCK)

```

```

(
    HUB_Customer_LDTS,
    HUB_Customer_RSRC,
    CustomerID
)
SELECT
    @BatchDate AS HUB_Customer_LDTS,
    'TPCE' AS HUB_Customer_RSRC,
    c.C_ID CustomerID
FROM
    dbo.CUSTOMER c WITH (NOLOCK)

```

```

SET @QueryEnd = SYSDATETIME();

```

```

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.HUB_Customer;

```

```

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_Customer',
    'HUB',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

```

GO

```

```

/*****
HUB_Company
*****/

```

```

USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.HUB_Company WITH (TABLOCK)
(
    HUB_Company_LDTS,
    HUB_Company_RSRC,
    CompanyName
)
SELECT
    @BatchDate AS HUB_Company_LDTS,
    'TPCE' AS HUB_Company_RSRC,
    co.CO_NAME CompanyName
FROM
    dbo.COMPANY co WITH (NOLOCK);

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.HUB_Company;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',

```

```
'HUB_Company',  
'HUB',  
@QueryStart,  
@QueryEnd,  
NULL,  
NULL,  
@CountTotal;
```

GO

```
/*  
HUB_Broker  
*/
```

```
USE TPCE;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DV.dbo.HUB_Broker WITH (TABLOCK)
```

```
(  
    HUB_Broker_LDTS,  
    HUB_Broker_RSRC,  
    BrokerID  
)
```

```
SELECT
```

```
    @BatchDate AS HUB_Broker_LDTS,  
    'TPCE' AS HUB_Broker_RSRC,  
    b.B_ID BrokerID
```

```
FROM
```

```
    dbo.[BROKER] b WITH (NOLOCK);
```

```
SET @QueryEnd = SYSDATETIME();
```

```
DECLARE
```

```
    @Iteration AS INT,  
    @CountTotal AS INT;
```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
```

```
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.HUB_Broker;
```

```
INSERT INTO master.dbo.LoadResults
```

```
SELECT
```

```
    @Iteration,  
    'DV',  
    'HUB_Broker',  
    'HUB',  
    @QueryStart,  
    @QueryEnd,  
    NULL,  
    NULL,  
    @CountTotal;
```

```
GO
```

```
/*  
HUB_Account  
*/
```

```
USE TPCE;
```

```
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
```

```

SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.HUB_Account WITH (TABLOCK)
(
    HUB_Account_LDTS,
    HUB_Account_RSRC,
    AccountID
)
SELECT
    @BatchDate AS HUB_Account_LDTS,
    'TPCE' AS HUB_Account_RSRC,
    ca.CA_ID AS AccountID
FROM
    dbo.CUSTOMER_ACCOUNT ca WITH (NOLOCK);

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.HUB_Account;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_Account',
    'HUB',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

```

/*****
XREF_Town
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.XREF_Town WITH (TABLOCK)
(
    PostalCode,
    XREF_Town_LDTS,
    XREF_Town_RSRC,
    TownName,
    StateProvinceCode,
    CountryName
)
SELECT DISTINCT
    zc.ZC_CODE AS PostalCode,
    @BatchDate AS XREF_Town_LDTS,
    'TPCE' AS XREF_Town_RSRC,
    zc.ZC_TOWN AS TownName,
    zc.ZC_DIV AS StateProvinceCode,
    ad.AD_CTRY AS CountryName
FROM
    dbo.[ADDRESS] ad WITH (NOLOCK)
    JOIN dbo.ZIP_CODE zc WITH (NOLOCK)
        ON zc.ZC_CODE = ad.AD_ZC_CODE;

SET @QueryEnd = SYSDATETIME();

DECLARE

```

```

        @Iteration AS INT,
        @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.XREF_Town;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'XREF_Town',
    'XREF',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

```

/*****
LNK_SecurityCompany
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.LNK_SecurityCompany WITH (TABLOCK)
(
    LNK_SecurityCompany_LDTS,

```



```

        LNK_SecurityCompany_RSRC,
        HUB_Security_SQN,
        HUB_Company_SQN
    )
SELECT
    @BatchDate AS LNK_SecurityCompany_LDTS,
    'TPCE' AS LNK_SecurityCompany_RSRC,
    hs.HUB_Security_SQN AS HUB_Security_SQN,
    hco.HUB_Company_SQN AS HUB_Company_SQN
FROM
    dbo.[SECURITY] s WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Security hs WITH (NOLOCK)
        ON hs.SecurityCode = s.S_SYMB
    JOIN dbo.COMPANY co WITH (NOLOCK)
        ON co.CO_ID = s.S_CO_ID
    JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
        ON hco.CompanyName = co.CO_NAME;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.LNK_SecurityCompany;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'LNK_SecurityCompany',
    'Link',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

```

/*****
LNK_AccountCustomer
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.LNK_AccountCustomer WITH (TABLOCK)
(
    LNK_AccountCustomer_LDTS,
    LNK_AccountCustomer_RSRC,
    HUB_Account_SQN,
    HUB_Customer_SQN
)
SELECT
    @BatchDate AS LNK_AccountCustomer_LDTS,
    'TPCE' AS LNK_AccountCustomer_RSRC,
    ha.HUB_Account_SQN AS HUB_Account_SQN,
    hc.HUB_Customer_SQN AS HUB_Customer_SQN
FROM
    dbo.CUSTOMER_ACCOUNT ca WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Account ha WITH (NOLOCK)
        ON ha.AccountID = ca.CA_ID
    JOIN TPCE_DV.dbo.HUB_Customer hc WITH (NOLOCK)
        ON hc.CustomerID = ca.CA_C_ID;

SET @QueryEnd = SYSDATETIME();

DECLARE

```

```

        @Iteration AS INT,
        @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.LNK_AccountCustomer;

```

```

INSERT INTO master.dbo.LoadResults
SELECT

```

```

    @Iteration,
    'DV',
    'LNK_AccountCustomer',
    'Link',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

```

GO

```

```

/*****
LNK_AccountBroker
*****/

```

```

USE TPCE;
GO

```

```

SET NOCOUNT ON;

```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

```

```

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

```

```

INSERT INTO TPCE_DV.dbo.LNK_AccountBroker WITH (TABLOCK)
(
    LNK_AccountBroker_LDTS,

```

```

        LNK_AccountBroker_RSRC,
        HUB_Account_SQN,
        HUB_Broker_SQN
    )
SELECT
    @BatchDate AS LNK_AccountBroker_LDTS,
    'TPCE' AS LNK_AccountBroker_RSRC,
    ha.HUB_Account_SQN AS HUB_Account_SQN,
    hb.HUB_Broker_SQN AS HUB_Broker_SQN
FROM
    dbo.CUSTOMER_ACCOUNT ca WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Account ha WITH (NOLOCK)
        ON ha.AccountID = ca.CA_ID
    JOIN TPCE_DV.dbo.HUB_Broker hb WITH (NOLOCK)
        ON hb.BrokerID = ca.CA_B_ID;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.LNK_AccountBroker;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'LNK_AccountBroker',
    'Link',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

```

/*****
LNK_SecurityExchange
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.LNK_SecurityExchange WITH (TABLOCK)
(
    LNK_SecurityExchange_LDTS,
    LNK_SecurityExchange_RSRC,
    HUB_Security_SQN,
    HUB_Exchange_SQN
)
SELECT
    @BatchDate AS LNK_SecurityExchange_LDTS,
    'TPCE' AS LNK_SecurityExchange_RSRC,
    hs.HUB_Security_SQN AS HUB_Security_SQN,
    hex.HUB_Exchange_SQN AS HUB_Exchange_SQN
FROM
    dbo.[SECURITY] s WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Security hs WITH (NOLOCK)
        ON hs.SecurityCode = s.S_SYMB
    JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)
        ON hex.ExchangeID = s.S_EX_ID;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.LNK_SecurityExchange;
```

```
INSERT INTO master.dbo.LoadResults
SELECT
```

```
    @Iteration,
    'DV',
    'LNK_SecurityExchange',
    'Link',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;
```

```
GO
```

```
/*
SAT_TradeType
*/
```

```
USE TPCE;
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DV.dbo.SAT_TradeType WITH (TABLOCK)
(
    HUB_TradeType_SQN,
    SAT_TradeType_LDTS,
    SAT_TradeType_RSRC,
```

```

        TradeTypeCode,
        IsSellFlag,
        IsMarketFlag
    )
SELECT
    htt.HUB_TradeType_SQN AS HUB_TradeType_SQN,
    @BatchDate AS SAT_TradeType_LDTS,
    'TPCE' AS SAT_TradeType_RSRC,
    tt.TT_ID AS TradeTypeCode,
    tt.TT_IS_SELL AS IsSellFlag,
    tt.TT_IS_MRKT AS IsMarketFlag
FROM
    dbo.TRADE_TYPE tt WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_TradeType htt WITH (NOLOCK)
        ON htt.TradeTypeDesc = tt.TT_NAME;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_TradeType;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_TradeType',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

```

/*****
SAT_Security
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.SAT_Security WITH (TABLOCK)
(
    HUB_Security_SQN,
    SAT_Security_LDTS,
    SAT_Security_RSRC,
    SecurityName,
    IssueType,
    FirstTradeDate,
    StatusCode
)
SELECT
    hs.HUB_Security_SQN AS HUB_Security_SQN,
    @BatchDate AS SAT_Security_LDTS,
    'TPCE' AS SAT_Security_RSRC,
    s.S_NAME AS SecurityName,
    s.S_ISSUE AS IssueType,
    s.S_START_DATE AS FirstTradeDate,
    s.S_ST_ID AS StatusCode
FROM
    dbo.[SECURITY] s WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Security hs WITH (NOLOCK)
        ON hs.SecurityCode = s.S_SYMB;

SET @QueryEnd = SYSDATETIME();

```



```

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_Security;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_Security',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

GO

```

/*****
SAT_ExchangeSecurities
*****/

```

```

USE TPCE;
GO

```

```

SET NOCOUNT ON;

```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

```

```

INSERT INTO TPCE_DV.dbo.SAT_ExchangeSecurities WITH (TABLOCK)

```

```

(
    HUB_Exchange_SQN,
    SAT_ExchangeSecurities_LDTS,
    SAT_ExchangeSecurities_RSRC,
    ListedSecurities
)
SELECT
    hex.HUB_Exchange_SQN AS HUB_Exchange_SQN,
    @BatchDate AS SAT_Exchange_LDTS,
    'TPCE' AS SAT_Exchange_RSRC,
    ex.EX_NUM_SYMB AS ListedSecurities
FROM
    dbo.EXCHANGE ex WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)
        ON hex.ExchangeID = ex.EX_ID;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_ExchangeSecurities;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_ExchangeSecurities',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

```

/*****
SAT_Exchange
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.SAT_Exchange WITH (TABLOCK)
(
    HUB_Exchange_SQN,
    SAT_Exchange_LDTS,
    SAT_Exchange_RSRC,
    ExchangeName,
    ExchangeDesc,
    OpeningTime,
    ClosingTime,
    AddressLine1,
    AddressLine2,
    PostalCode
)
SELECT
    hex.HUB_Exchange_SQN AS HUB_Exchange_SQN,
    @BatchDate AS SAT_Exchange_LDTS,
    'TPCE' AS SAT_Exchange_RSRC,
    ex.EX_NAME AS ExchangeName,
    ex.EX_DESC AS ExchangeDesc,
    ex.EX_OPEN AS OpeningTime,
    ex.EX_CLOSE AS ClosingTime,
    ad.AD_LINE1 AS AddressLine1,
    ad.AD_LINE2 AS AddressLine2,
    ad.AD_ZC_CODE AS PostalCode

```

```

FROM
    dbo.EXCHANGE ex WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)
        ON hex.ExchangeID = ex.EX_ID
    JOIN dbo.[ADDRESS] ad WITH (NOLOCK)
        ON ad.AD_ID = ex.EX_AD_ID;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_Exchange;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_Exchange',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
SAT_CustomerContact
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE

```

```
@BatchDate AS DATETIME2(0),  
@QueryStart AS DATETIME2(7),  
@QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DV.dbo.SAT_CustomerContact WITH (TABLOCK)
```

```
(  
    HUB_Customer_SQN,  
    SAT_CustomerContact_LDTS,  
    SAT_CustomerContact_RSRC,  
    PhoneCountryCodePrimary,  
    PhoneAreaCodePrimary,  
    PhoneLocalNumberPrimary,  
    PhoneExtensionPrimary,  
    PhoneCountryCodeSecondary,  
    PhoneAreaCodeSecondary,  
    PhoneLocalNumberSecondary,  
    PhoneExtensionSecondary,  
    EmailPrimary,  
    EmailSecondary,  
    AddressLine1,  
    AddressLine2,  
    PostalCode  
)
```

```
SELECT
```

```
hc.HUB_Customer_SQN AS HUB_Customer_SQN,  
@BatchDate AS SAT_Customer_LDTS,  
'TPCE' AS SAT_Customer_RSRC,  
c.C_CTRY_1 AS PhoneCountryCodePrimary,  
c.C_AREA_1 AS PhoneAreaCodePrimary,  
c.C_LOCAL_1 AS PhoneLocalNumberPrimary,  
c.C_EXT_1 AS PhoneExtensionPrimary,  
c.C_CTRY_2 AS PhoneCountryCodeSecondary,  
c.C_AREA_2 AS PhoneAreaCodeSecondary,  
c.C_LOCAL_2 AS PhoneLocalNumberSecondary,  
c.C_EXT_2 AS PhoneExtensionSecondary,  
c.C_EMAIL_1 AS EmailPrimary,  
c.C_EMAIL_2 AS EmailSecondary,
```

```

        ad.AD_LINE1 AS AddressLine1,
        ad.AD_LINE2 AS AddressLine2,
        ad.AD_ZC_CODE AS PostalCode
FROM
    dbo.CUSTOMER c WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Customer hc WITH (NOLOCK)
        ON hc.CustomerID = c.C_ID
    JOIN dbo.[ADDRESS] ad WITH (NOLOCK)
        ON ad.AD_ID = c.C_AD_ID;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_CustomerContact;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_CustomerContact',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
SAT_Customer
*****/
USE TPCE;
GO

```

```

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.SAT_Customer WITH (TABLOCK)
(
    HUB_Customer_SQN,
    SAT_Customer_LDTS,
    SAT_Customer_RSRC,
    LastName,
    FirstName,
    MiddleInitial,
    GENDER,
    DateOfBirth,
    TierNumber,
    StatusCode
)
SELECT
    hc.HUB_Customer_SQN AS HUB_Customer_SQN,
    @BatchDate AS SAT_Customer_LDTS,
    'TPCE' AS SAT_Customer_RSRC,
    c.C_L_NAME AS LastName,
    c.C_F_NAME AS FirstName,
    c.C_M_NAME AS MiddleInitial,
    c.C_GNDR AS GENDER,
    c.C_DOB AS DateOfBirth,
    c.C_TIER AS TierNumber,
    c.C_ST_ID AS StatusCode
FROM
    dbo.CUSTOMER c WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Customer hc WITH (NOLOCK)
        ON hc.CustomerID = c.C_ID

SET @QueryEnd = SYSDATETIME();

```

```

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_Customer;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_Customer',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

GO

```

/*****
SAT_CompanyCreditRating
*****/

```

```

USE TPCE;
GO

```

```

SET NOCOUNT ON;

```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

```

```

INSERT INTO TPCE_DV.dbo.SAT_CompanyCreditRating WITH (TABLOCK)

```



```

(
    HUB_Company_SQN,
    SAT_CompanyCreditRating_LDTS,
    SAT_CompanyCreditRating_RSRC,
    CompanyCreditRating
)
SELECT
    hco.HUB_Company_SQN AS HUB_Company_SQN,
    @BatchDate AS SAT_CompanyCreditRating_LDTS,
    'TPCE' AS SAT_CompanyCreditRating_RSRC,
    co.CO_SP_RATE AS CompanyCreditRating
FROM
    dbo.COMPANY co WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
        ON hco.CompanyName = co.CO_NAME;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_CompanyCreditRating;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_CompanyCreditRating',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

```

/*****
SAT_CompanyCEO
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.SAT_CompanyCEO WITH (TABLOCK)
(
    HUB_Company_SQN,
    SAT_CompanyCEO_LDTS,
    SAT_CompanyCEO_RSRC,
    CompanyCEO
)
SELECT
    hco.HUB_Company_SQN AS HUB_Company_SQN,
    @BatchDate AS SAT_CompanyCEO_LDTS,
    'TPCE' AS SAT_CompanyCEO_RSRC,
    co.CO_CEO AS CompanyCEO
FROM
    dbo.COMPANY co WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
        ON hco.CompanyName = co.CO_NAME;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;

```

```
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_CompanyCEO;
```

```
INSERT INTO master.dbo.LoadResults
```

```
SELECT
```

```
    @Iteration,  
    'DV',  
    'SAT_CompanyCEO',  
    'Satellite',  
    @QueryStart,  
    @QueryEnd,  
    NULL,  
    NULL,  
    @CountTotal;
```

```
GO
```

```
/*  
SAT_Company  
*/
```

```
USE TPCE;
```

```
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
```

```
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DV.dbo.SAT_Company WITH (TABLOCK)
```

```
(
```

```
    HUB_Company_SQN,  
    SAT_Company_LDTS,  
    SAT_Company_RSRC,  
    CompanyID,  
    CompanyDesc,
```

```

        CompanyOpenDate,
        AddressLine1,
        AddressLine2,
        PostalCode,
        IndustryCode,
        SectorCode,
        StatusCode
    )
SELECT
    hco.HUB_Company_SQN AS HUB_Company_SQN,
    @BatchDate AS SAT_Company_LDTS,
    'TPCE' AS SAT_Company_RSRC,
    co.CO_ID AS CompanyID,
    co.CO_DESC AS CompanyDesc,
    co.CO_OPEN_DATE AS CompanyOpenDate,
    ad.AD_LINE1 AS AddressLine1,
    ad.AD_LINE2 AS AddressLine2,
    ad.AD_ZC_CODE AS PostalCode,
    ind.IN_ID AS IndustryCode,
    ind.IN_SC_ID AS SectorCode,
    co.CO_ST_ID AS StatusCode
FROM
    dbo.COMPANY co WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
        ON hco.CompanyName = co.CO_NAME
    JOIN dbo.[ADDRESS] ad WITH (NOLOCK)
        ON ad.AD_ID = co.CO_AD_ID
    JOIN dbo.INDUSTRY ind WITH (NOLOCK)
        ON ind.IN_ID = co.CO_IN_ID;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_Company;

INSERT INTO master.dbo.LoadResults

```

```
SELECT
    @Iteration,
    'DV',
    'SAT_Company',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;
```

```
GO
```

```
/******
SAT_Broker
******/
```

```
USE TPCE;
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DV.dbo.SAT_Broker WITH (TABLOCK)
(
    HUB_Broker_SQN,
    SAT_Broker_LDTS,
    SAT_Broker_RSRC,
    BrokerName,
    StatusCode
)
```

```
SELECT
    hb.HUB_Broker_SQN AS HUB_Broker_SQN,
```

```

        @BatchDate AS SAT_Broker_LDTS,
        'TPCE' AS SAT_Broker_RSRC,
        b.B_NAME AS BrokerName,
        b.B_ST_ID AS StatusCode
FROM
    dbo.[BROKER] b WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Broker hb WITH (NOLOCK)
        ON hb.BrokerID = b.B_ID;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_Broker;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_Broker',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
SAT_Account
*****/
USE TPCE;
GO

SET NOCOUNT ON;

```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.SAT_Account WITH (TABLOCK)
(
    HUB_Account_SQN,
    SAT_Account_LDTS,
    SAT_Account_RSRC,
    AccountName,
    TaxStatusCode
)
SELECT
    ha.HUB_Account_SQN AS HUB_Account_SQN,
    @BatchDate AS SAT_Account_LDTS,
    'TPCE' AS SAT_Account_RSRC,
    ca.CA_NAME AS AccountName,
    ca.CA_TAX_ST AS TaxStatusCode
FROM
    dbo.CUSTOMER_ACCOUNT ca WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Account ha WITH (NOLOCK)
        ON ha.AccountID = ca.CA_ID;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_Account;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,

```

```
'DV',  
'SAT_Account',  
'Satellite',  
@QueryStart,  
@QueryEnd,  
NULL,  
NULL,  
@CountTotal;
```

GO

```
/*  
SAT_SecurityCompanyStatus  
*/
```

```
USE TPCE;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DV.dbo.SAT_SecurityCompanyStatus WITH (TABLOCK)
```

```
(  
    LNK_SecurityCompany_SQN,  
    SAT_SecurityCompanyStatus_LDTS,  
    SAT_SecurityCompanyStatus_RSRC  
)
```

```
SELECT
```

```
    lsc.LNK_SecurityCompany_SQN AS LNK_SecurityCompany_SQN,  
    @BatchDate AS SAT_AccountBrokerStatus_LDTS,  
    'TPCE' AS SAT_AccountBrokerStatus_RSRC
```

```
FROM
```

```
    TPCE_DV.dbo.LNK_SecurityCompany lsc WITH (NOLOCK);
```



```

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_SecurityCompanyStatus;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_SecurityCompanyStatus',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

```

```
GO
```

```

/*****
SAT_AccountCustomerStatus
*****/

```

```

USE TPCE;
GO

```

```
SET NOCOUNT ON;
```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

```

```

INSERT INTO TPCE_DV.dbo.SAT_AccountCustomerStatus WITH (TABLOCK)
(
    LNK_AccountCustomer_SQN,
    SAT_AccountCustomerStatus_LDTS,
    SAT_AccountCustomerStatus_RSRC
)
SELECT
    lab.LNK_AccountCustomer_SQN AS LNK_AccountCustomer_SQN,
    @BatchDate AS SAT_AccountCustomerStatus_LDTS,
    'TPCE' AS SAT_AccountCustomerStatus_RSRC
FROM
    TPCE_DV.dbo.LNK_AccountCustomer lab WITH (NOLOCK);

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_AccountCustomerStatus;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_AccountCustomerStatus',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
SAT_AccountBrokerStatus

```

```

*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

INSERT INTO TPCE_DV.dbo.SAT_AccountBrokerStatus WITH (TABLOCK)
(
    LNK_AccountBroker_SQN,
    SAT_AccountBrokerStatus_LDTS,
    SAT_AccountBrokerStatus_RSRC
)
SELECT
    lab.LNK_AccountBroker_SQN AS LNK_AccountBroker_SQN,
    @BatchDate AS SAT_AccountBrokerStatus_LDTS,
    'TPCE' AS SAT_AccountBrokerStatus_RSRC
FROM
    TPCE_DV.dbo.LNK_AccountBroker lab WITH (NOLOCK);

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_AccountBrokerStatus;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',

```

```
'SAT_AccountBrokerStatus',  
'Satellite',  
@QueryStart,  
@QueryEnd,  
NULL,  
NULL,  
@CountTotal;
```

GO

```
/*  
SAT_SecurityExchangeStatus  
*/
```

```
USE TPCE;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = '2005/01/17 22:15:27';
```

```
INSERT INTO TPCE_DV.dbo.SAT_SecurityExchangeStatus WITH (TABLOCK)
```

```
(  
    LNK_SecurityExchange_SQN,  
    SAT_SecurityExchangeStatus_LDTS,  
    SAT_SecurityExchangeStatus_RSRC  
)
```

```
SELECT
```

```
    lse.LNK_SecurityExchange_SQN AS LNK_SecurityExchange_SQN,  
    @BatchDate AS SAT_AccountBrokerStatus_LDTS,  
    'TPCE' AS SAT_AccountBrokerStatus_RSRC
```

```
FROM
```

```
    TPCE_DV.dbo.LNK_SecurityExchange lse WITH (NOLOCK);
```

```

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.SAT_SecurityExchangeStatus;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_SecurityExchangeStatus',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
TLNK_QuarterlyCompanyPerformance
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

```

```
INSERT INTO TPCE_DV.dbo.TLNK_QuarterlyCompanyPerformance WITH (TABLOCK)
```

```
(
```

```
    HUB_Company_SQN,  
    FiscalQuarter,  
    FiscalYear,  
    TLNK_QuarterlyCompanyPerformance_LDTS,  
    TLNK_QuarterlyCompanyPerformance_RSRC,  
    FiscalQuarterStartDate,  
    Revenue,  
    NetEarnings,  
    EarningsPerShareBasic,  
    EarningsPerShareDiluted,  
    Margin,  
    Inventory,  
    Assets,  
    Liabilities,  
    SharesOutstandingBasicAvg,  
    SharesOutstandingDilutedAvg
```

```
)
```

```
SELECT
```

```
    hco.HUB_Company_SQN AS HUB_Company_SQN,  
    fi.FI_QTR AS FiscalQuarter,  
    fi.FI_YEAR AS FiscalYear,  
    @BatchDate AS TLNK_QuarterlyCompanyPerformance_LDTS,  
    'TPCE' AS TLNK_QuarterlyCompanyPerformance_RSRC,  
    fi.FI_QTR_START_DATE AS FiscalQuarterStartDate,  
    fi.FI_REVENUE AS Revenue,  
    fi.FI_NET_EARN AS NetEarnings,  
    fi.FI_BASIC_EPS AS EarningsPerShareBasic,  
    fi.FI_DILUT_EPS AS EarningsPerShareDiluted,  
    fi.FI_MARGIN AS Margin,  
    fi.FI_INVENTORY AS Inventory,  
    fi.FI_ASSETS AS Assets,  
    fi.FI_LIABILITY AS Liabilities,  
    fi.FI_OUT_BASIC AS SharesOutstandingBasicAvg,  
    fi.FI_OUT_DILUT AS SharesOutstandingDilutedAvg
```

```
FROM
```

```
    dbo.FINANCIAL fi WITH (NOLOCK)  
    JOIN dbo.COMPANY co WITH (NOLOCK)  
        ON co.CO_ID = fi.FI_CO_ID
```

```

JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
    ON hco.CompanyName = co.CO_NAME;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.TLNK_QuarterlyCompanyPerformance;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'TLNK_QuarterlyCompanyPerformance',
    'Transactional',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
TLNK_DailyMarketPerformance
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

```

```

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';

ALTER INDEX IDX_TLNK_DailyMarketPerformance_HUB_Company_SQN ON TPCE_DV.dbo.TLNK_DailyMarketPerformance DISABLE;

INSERT INTO TPCE_DV.dbo.TLNK_DailyMarketPerformance WITH (TABLOCK)
(
    HUB_Security_SQN,
    [Date],
    TLNK_DailyMarketPerformance_LDTS,
    TLNK_DailyMarketPerformance_RSRC,
    HUB_Company_SQN,
    HUB_Exchange_SQN,
    DayClose,
    DayHigh,
    DayLow,
    DayVolume
)
SELECT
    hs.HUB_Security_SQN AS HUB_Security_SQN,
    dm.DM_DATE AS [Date],
    @BatchDate AS TLNK_DailyMarketPerformance_LDTS,
    'TPCE' AS TLNK_DailyMarketPerformance_RSRC,
    hco.HUB_Company_SQN AS HUB_Company_SQN,
    hex.HUB_Exchange_SQN AS HUB_Exchange_SQN,
    dm.DM_CLOSE AS DayClose,
    dm.DM_HIGH AS DayHigh,
    dm.DM_LOW AS DayLow,
    dm.DM_VOL AS DayVolume
FROM
    dbo.DAILY_MARKET dm WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Security hs WITH (NOLOCK)
        ON hs.SecurityCode = dm.DM_S_SYMB
    JOIN dbo.[SECURITY] s WITH (NOLOCK)
        ON s.S_SYMB = dm.DM_S_SYMB
    JOIN dbo.COMPANY co WITH (NOLOCK)
        ON co.CO_ID = s.S_CO_ID
    JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
        ON hco.CompanyName = co.CO_NAME
    JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)

```



```

        ON hex.ExchangeID = s.S_EX_ID
WHERE
    dm.DM_DATE >= '2004/01/01';

ALTER INDEX IDX_TLNK_DailyMarketPerformance_HUB_Company_SQN ON TPCE_DV.dbo.TLNK_DailyMarketPerformance REBUILD;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.TLNK_DailyMarketPerformance;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'TLNK_DailyMarketPerformance',
    'Transactional',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

/*****
TLNK_Trade
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),

```

```

@QueryStart AS DATETIME2(7),
@QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = '2005/01/17 22:15:27';
ALTER INDEX IDX_TLNK_Trade_HUB_Account_SQN ON TPCE_DV.dbo.TLNK_Trade DISABLE;
ALTER INDEX IDX_TLNK_Trade_HUB_Broker_SQN ON TPCE_DV.dbo.TLNK_Trade DISABLE;
ALTER INDEX IDX_TLNK_Trade_HUB_Company_SQN ON TPCE_DV.dbo.TLNK_Trade DISABLE;
ALTER INDEX IDX_TLNK_Trade_HUB_Customer_SQN ON TPCE_DV.dbo.TLNK_Trade DISABLE;
ALTER INDEX IDX_TLNK_Trade_HUB_Security_SQN ON TPCE_DV.dbo.TLNK_Trade DISABLE;

INSERT INTO TPCE_DV.dbo.TLNK_Trade WITH (TABLOCK)
(
    TradeID,
    TLNK_Trade_LDTS,
    TLNK_Trade_RSRC,
    HUB_Broker_SQN,
    HUB_Customer_SQN,
    HUB_Account_SQN,
    HUB_Company_SQN,
    HUB_Exchange_SQN,
    HUB_Security_SQN,
    HUB_TradeType_SQN,
    TradeDateTime,
    Quantity,
    BidPrice,
    TradePrice,
    Charge,
    Commission,
    Tax,
    IsCashFlag,
    StatusCode
)
SELECT
    t.T_ID AS TradeID,
    @BatchDate AS TLNK_Trade_LDTS,
    'TPCE' AS TLNK_Trade_RSRC,
    hb.HUB_Broker_SQN AS HUB_Broker_SQN,
    hc.HUB_Customer_SQN AS HUB_Customer_SQN,
    ha.HUB_Account_SQN AS HUB_Account_SQN,

```

```

hco.HUB_Company_SQN AS HUB_Company_SQN,
hex.HUB_Exchange_SQN AS HUB_Exchange_SQN,
hs.HUB_Security_SQN AS HUB_Security_SQN,
htt.HUB_TradeType_SQN AS HUB_TradeType_SQN,
t.T_DTS AS TradeDateTime,
t.T_QTY AS Quantity,
t.T_BID_PRICE AS BidPrice,
t.T_TRADE_PRICE AS TradePrice,
t.T_CHRG AS Charge,
t.T_COMM AS Commission,
t.T_TAX AS Tax,
t.T_IS_CASH AS IsCashFlag,
t.T_ST_ID AS StatusCode

```

FROM

```

dbo.TRADE t WITH (NOLOCK)
JOIN dbo.CUSTOMER_ACCOUNT ca WITH (NOLOCK)
    ON ca.CA_ID = t.T_CA_ID
JOIN TPCE_DV.dbo.HUB_Broker hb WITH (NOLOCK)
    ON hb.BrokerID = ca.CA_B_ID
JOIN TPCE_DV.dbo.HUB_Customer hc WITH (NOLOCK)
    ON hc.CustomerID = ca.CA_C_ID
JOIN TPCE_DV.dbo.HUB_Account ha WITH (NOLOCK)
    ON ha.AccountID = ca.CA_ID
JOIN dbo.[SECURITY] s WITH (NOLOCK)
    ON s.S_SYMB = t.T_S_SYMB
JOIN dbo.COMPANY co WITH (NOLOCK)
    ON co.CO_ID = s.S_CO_ID
JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
    ON hco.CompanyName = co.CO_NAME
JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)
    ON hex.ExchangeID = s.S_EX_ID
JOIN TPCE_DV.dbo.HUB_Security hs WITH (NOLOCK)
    ON hs.SecurityCode = s.S_SYMB
JOIN dbo.TRADE_TYPE tt WITH (NOLOCK)
    ON tt.TT_ID = t.T_TT_ID
JOIN TPCE_DV.dbo.HUB_TradeType htt WITH (NOLOCK)
    ON htt.TradeTypeDesc = tt.TT_NAME;

```

```

ALTER INDEX IDX_TLNK_Trade_HUB_Account_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;
ALTER INDEX IDX_TLNK_Trade_HUB_Broker_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;

```

```

ALTER INDEX IDX_TLNK_Trade_HUB_Company_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;
ALTER INDEX IDX_TLNK_Trade_HUB_Customer_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;
ALTER INDEX IDX_TLNK_Trade_HUB_Security_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.TLNK_Trade;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'TLNK_Trade',
    'Transactional',
    @QueryStart,
    @QueryEnd,
    NULL,
    NULL,
    @CountTotal;

GO

```

## 1.2.2 Incremental Data Loading Scripts

### 1.2.2.1 TPC Incremental Data Import T-SQL Script

```

USE TPCE;
GO

```

```

BULK INSERT dbo.[ADDRESS] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Address.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.[BROKER] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Broker.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.COMPANY FROM 'C:\Users\Administrator\Desktop\Source Data\History\Company.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.CUSTOMER FROM 'C:\Users\Administrator\Desktop\Source Data\History\Customer.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.CUSTOMER_ACCOUNT FROM 'C:\Users\Administrator\Desktop\Source Data\History\CustomerAccount.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.DAILY_MARKET FROM 'C:\Users\Administrator\Desktop\Source Data\History\DailyMarket.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.EXCHANGE FROM 'C:\Users\Administrator\Desktop\Source Data\History\Exchange.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.FINANCIAL FROM 'C:\Users\Administrator\Desktop\Source Data\History\Financial.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.INDUSTRY FROM 'C:\Users\Administrator\Desktop\Source Data\History\Industry.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.SECTOR FROM 'C:\Users\Administrator\Desktop\Source Data\History\Sector.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.[SECURITY] FROM 'C:\Users\Administrator\Desktop\Source Data\History\Security.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.STATUS_TYPE FROM 'C:\Users\Administrator\Desktop\Source Data\History\StatusType.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.TRADE FROM 'C:\Users\Administrator\Desktop\Source Data\History\Trade.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.TRADE_TYPE FROM 'C:\Users\Administrator\Desktop\Source Data\History\TradeType.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')
BULK INSERT dbo.ZIP_CODE FROM 'C:\Users\Administrator\Desktop\Source Data\History\ZipCode.txt'
    WITH (FIELDTERMINATOR = '|', ROWTERMINATOR = '\n')

```

### 1.2.2.2 TPCE\_DM Incremental Data Loading T-SQL Script

```

/*****
D_Broker
*****/
USE TPCE_DM;
GO

```

```

USE TPCE_DM;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.D_Broker AS dest
USING dbo.V_D_Broker AS src WITH (NOLOCK)
    ON dest.BrokerID = src.BrokerID
WHEN NOT MATCHED THEN
    INSERT VALUES
        (
            src.BrokerID,
            src.BrokerName,
            src.BrokerStatusCode,
            src.BrokerStatusDesc,
            @BatchDate,
            NULL
        )
WHEN MATCHED
    AND (dest.BrokerName != src.BrokerName
    OR dest.BrokerStatusCode != src.BrokerStatusCode
    OR dest.BrokerStatusDesc != src.BrokerStatusDesc) THEN
    UPDATE SET
        dest.BrokerName = src.BrokerName,
        dest.BrokerStatusCode = src.BrokerStatusCode,
        dest.BrokerStatusDesc = src.BrokerStatusDesc,
        dest.WH_DateUpdated = @BatchDate;

SET @QueryEnd = SYSDATETIME();

DECLARE

```

```

        @Iteration AS INT,
        @CountTotal AS INT,
        @CountInsert AS INT,
        @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM D_Broker;
SELECT @CountInsert = COUNT(*) FROM D_Broker WHERE WH_DateInserted = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM D_Broker WHERE WH_DateUpdated = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_Broker',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

/*****
D_CashType
*****/
USE TPCE_DM;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

```

```

MERGE dbo.D_CashType dest
USING dbo.V_D_CashType src WITH (NOLOCK)
    ON dest.IsCashFlag = src.IsCashFlag
WHEN NOT MATCHED THEN
    INSERT
    (
        CashType,
        IsCashFlag,
        WH_DateInserted
    )
VALUES
    (
        src.CashType,
        src.IsCashFlag,
        @BatchDate
    )
WHEN MATCHED AND dest.CashType != src.CashType THEN
    UPDATE
    SET dest.CashType = src.CashType,
        dest.WH_DateUpdated = @BatchDate;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM D_CashType;
SELECT @CountInsert = COUNT(*) FROM D_CashType WHERE WH_DateInserted = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM D_CashType WHERE WH_DateUpdated = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_CashType',

```



```
'Dimension',  
@QueryStart,  
@QueryEnd,  
@CountInsert,  
@CountUpdate,  
@CountTotal;
```

```
GO
```

```
/*  
D_Company  
*/
```

```
USE TPCE_DM;
```

```
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
```

```
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);
```

```
INSERT INTO dbo.D_Company WITH (TABLOCK)
```

```
(
```

```
    CompanyID,  
    CompanyName,  
    CompanyDesc,  
    CompanyStatusCode,  
    CompanyStatusDesc,  
    CompanyCEO,  
    CompanyCreditRating,  
    CompanyOpenDate,  
    IndustryCode,  
    IndustryDesc,  
    SectorCode,  
    SectorDesc,  
    AddressLine1,
```

```

        AddressLine2,
        TownName,
        StateProvinceCode,
        PostalCode,
        CountryName,
        WH_StartDate,
        WH_CurrInd
    )
SELECT
    mrg.CompanyID,
    mrg.CompanyName,
    mrg.CompanyDesc,
    mrg.CompanyStatusCode,
    mrg.CompanyStatusDesc,
    mrg.CompanyCEO,
    mrg.CompanyCreditRating,
    mrg.CompanyOpenDate,
    mrg.IndustryCode,
    mrg.IndustryDesc,
    mrg.SectorCode,
    mrg.SectorDesc,
    mrg.AddressLine1,
    mrg.AddressLine2,
    mrg.TownName,
    mrg.StateProvinceCode,
    mrg.PostalCode,
    mrg.CountryName,
    @BatchDate,
    1
FROM (
    MERGE dbo.D_Company AS dest
    USING dbo.V_D_Company AS src WITH (NOLOCK)
        ON dest.CompanyID = src.CompanyID
        AND dest.WH_CurrInd = 1
    WHEN NOT MATCHED THEN
        INSERT VALUES
        (
            src.CompanyID,
            src.CompanyName,
            src.CompanyDesc,

```

```

        src.CompanyStatusCode,
        src.CompanyStatusDesc,
        src.CompanyCEO,
        src.CompanyCreditRating,
        src.CompanyOpenDate,
        src.IndustryCode,
        src.IndustryDesc,
        src.SectorCode,
        src.SectorDesc,
        src.AddressLine1,
        src.AddressLine2,
        src.TownName,
        src.StateProvinceCode,
        src.PostalCode,
        src.CountryName,
        @BatchDate,
        NULL,
        1
    )
WHEN MATCHED
    AND dest.CompanyCEO != src.CompanyCEO
    OR (dest.CompanyCreditRating != src.CompanyCreditRating) THEN
    UPDATE SET
        dest.WH_EndDate = DATEADD(SECOND, -1, @BatchDate),
        dest.WH_CurrInd = 0
OUTPUT
    $ACTION MergeAction,
    src.CompanyID,
    src.CompanyName,
    src.CompanyDesc,
    src.CompanyStatusCode,
    src.CompanyStatusDesc,
    src.CompanyCEO,
    src.CompanyCreditRating,
    src.CompanyOpenDate,
    src.IndustryCode,
    src.IndustryDesc,
    src.SectorCode,
    src.SectorDesc,
    src.AddressLine1,

```

```

        src.AddressLine2,
        src.TownName,
        src.StateProvinceCode,
        src.PostalCode,
        src.CountryName
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

MERGE dbo.D_Company AS dest
USING dbo.V_D_Company AS src
    ON dest.CompanyID = src.CompanyID
    AND dest.WH_CurrInd = 1
WHEN MATCHED
    AND (dest.CompanyName != src.CompanyName
    OR dest.CompanyDesc != src.CompanyDesc
    OR dest.CompanyStatusCode != src.CompanyStatusCode
    OR dest.CompanyStatusDesc != src.CompanyStatusDesc
    OR dest.CompanyOpenDate != src.CompanyOpenDate
    OR dest.IndustryCode != src.IndustryCode
    OR dest.IndustryDesc != src.IndustryDesc
    OR dest.SectorCode != src.SectorCode
    OR dest.SectorDesc != src.SectorDesc
    OR dest.AddressLine1 != src.AddressLine1
    OR dest.AddressLine2 != src.AddressLine2
    OR dest.TownName != src.TownName
    OR dest.StateProvinceCode != src.StateProvinceCode
    OR dest.PostalCode != src.PostalCode
    OR dest.CountryName != src.CountryName) THEN
    UPDATE SET
        dest.CompanyName = src.CompanyName,
        dest.CompanyDesc = src.CompanyDesc,
        dest.CompanyStatusCode = src.CompanyStatusCode,
        dest.CompanyStatusDesc = src.CompanyStatusDesc,
        dest.CompanyOpenDate = src.CompanyOpenDate,
        dest.IndustryCode = src.IndustryCode,
        dest.IndustryDesc = src.IndustryDesc,
        dest.SectorCode = src.SectorCode,
        dest.SectorDesc = src.SectorDesc,
        dest.AddressLine1 = src.AddressLine1,

```

```
dest.AddressLine2 = src.AddressLine2,  
dest.TownName = src.TownName,  
dest.StateProvinceCode = src.StateProvinceCode,  
dest.PostalCode = src.PostalCode,  
dest.CountryName = src.CountryName;
```

```
SET @QueryEnd = SYSDATETIME();
```

```
DECLARE
```

```
    @Iteration AS INT,  
    @CountTotal AS INT,  
    @CountInsert AS INT,  
    @CountUpdate AS INT;
```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;  
SELECT @CountTotal = COUNT(*) FROM D_Company;  
SELECT @CountInsert = COUNT(*) FROM D_Company WHERE WH_StartDate = @BatchDate;  
SELECT @CountUpdate = COUNT(*) FROM D_Company WHERE WH_EndDate = DATEADD(SECOND,-1,@BatchDate);
```

```
INSERT INTO master.dbo.LoadResults
```

```
SELECT
```

```
    @Iteration,  
    'DM',  
    'D_Company',  
    'Dimension',  
    @QueryStart,  
    @QueryEnd,  
    @CountInsert,  
    @CountUpdate,  
    @CountTotal;
```

```
GO
```

```
/*  
D_CustomerAccount  
*/  
USE TPCE_DM;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
DECLARE @MergeOutput TABLE
```

```
(  
    MergeAction VARCHAR(10),  
    AccountID BIGINT,  
    AccountName VARCHAR(50),  
    TaxStatusCode TINYINT,  
    TaxStatusDesc VARCHAR(30),  
    CustomerID BIGINT,  
    LastName VARCHAR(25),  
    FirstName VARCHAR(20),  
    MiddleInitial CHAR(1),  
    GENDER VARCHAR(7),  
    DateOfBirth DATE,  
    TierNumber TINYINT,  
    TierDesc VARCHAR(35),  
    PhonePrimary VARCHAR(30),  
    PhoneSecondary VARCHAR(30),  
    EmailPrimary VARCHAR(50),  
    EmailSecondary VARCHAR(50),  
    AddressLine1 VARCHAR(80),  
    AddressLine2 VARCHAR(80),  
    TownName VARCHAR(80),  
    StateProvinceCode VARCHAR(80),  
    PostalCode VARCHAR(12),  
    CountryName VARCHAR(80),  
    CustomerStatusCode VARCHAR(4),  
    CustomerStatusDesc VARCHAR(10)  
)
```

```
SET @QueryStart = SYSDATETIME();
```

```
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);
```

```
MERGE dbo.D_CustomerAccount AS dest
```

```

USING dbo.V_D_CustomerAccount AS src WITH (NOLOCK)
    ON dest.AccountID = src.AccountID
    AND dest.WH_CurrInd = 1
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
        src.AccountID,
        src.AccountName,
        src.TaxStatusCode,
        src.TaxStatusDesc,
        src.CustomerID,
        src.LastName,
        src.FirstName,
        src.MiddleInitial,
        src.GENDER,
        src.DateOfBirth,
        src.TierNumber,
        src.TierDesc,
        src.PhonePrimary,
        src.PhoneSecondary,
        src.EmailPrimary,
        src.EmailSecondary,
        src.AddressLine1,
        src.AddressLine2,
        src.TownName,
        src.StateProvinceCode,
        src.PostalCode,
        src.CountryName,
        src.CustomerStatusCode,
        src.CustomerStatusDesc,
        @BatchDate,
        NULL,
        1
    )
WHEN MATCHED
    AND (dest.TierNumber != src.TierNumber
    OR dest.TierDesc != src.TierDesc) THEN
    UPDATE SET
        dest.WH_EndDate = DATEADD(SECOND, -1, @BatchDate),
        dest.WH_CurrInd = 0

```

## OUTPUT

```
$ACTION MergeAction,  
src.AccountID,  
src.AccountName,  
src.TaxStatusCode,  
src.TaxStatusDesc,  
src.CustomerID,  
src.LastName,  
src.FirstName,  
src.MiddleInitial,  
src.GENDER,  
src.DateOfBirth,  
src.TierNumber,  
src.TierDesc,  
src.PhonePrimary,  
src.PhoneSecondary,  
src.EmailPrimary,  
src.EmailSecondary,  
src.AddressLine1,  
src.AddressLine2,  
src.TownName,  
src.StateProvinceCode,  
src.PostalCode,  
src.CountryName,  
src.CustomerStatusCode,  
src.CustomerStatusDesc
```

## INTO

```
@MergeOutput;
```

```
INSERT INTO dbo.D_CustomerAccount WITH (TABLOCK)
```

```
(
```

```
AccountID,  
AccountName,  
TaxStatusCode,  
TaxStatusDesc,  
CustomerID,  
LastName,  
FirstName,  
MiddleInitial,  
GENDER,
```



```
DateOfBirth,  
TierNumber,  
TierDesc,  
PhonePrimary,  
PhoneSecondary,  
EmailPrimary,  
EmailSecondary,  
AddressLine1,  
AddressLine2,  
TownName,  
StateProvinceCode,  
PostalCode,  
CountryName,  
CustomerStatusCode,  
CustomerStatusDesc,  
WH_StartDate,  
WH_CurrInd
```

```
)
```

```
SELECT
```

```
mrg.AccountID,  
mrg.AccountName,  
mrg.TaxStatusCode,  
mrg.TaxStatusDesc,  
mrg.CustomerID,  
mrg.LastName,  
mrg.FirstName,  
mrg.MiddleInitial,  
mrg.GENDER,  
mrg.DateOfBirth,  
mrg.TierNumber,  
mrg.TierDesc,  
mrg.PhonePrimary,  
mrg.PhoneSecondary,  
mrg.EmailPrimary,  
mrg.EmailSecondary,  
mrg.AddressLine1,  
mrg.AddressLine2,  
mrg.TownName,  
mrg.StateProvinceCode,  
mrg.PostalCode,
```

```

        mrg.CountryName,
        mrg.CustomerStatusCode,
        mrg.CustomerStatusDesc,
        @BatchDate,
        1
FROM
    @MergeOutput mrg
WHERE
    mrg.MergeAction = 'UPDATE';

MERGE dbo.D_CustomerAccount AS dest
USING dbo.V_D_CustomerAccount AS src WITH (NOLOCK)
    ON dest.AccountID = src.AccountID
    AND dest.WH_CurrInd = 1
WHEN MATCHED
    AND (dest.AccountName != src.AccountName
    OR dest.TaxStatusCode != src.TaxStatusCode
    OR dest.TaxStatusDesc != src.TaxStatusDesc
    OR dest.CustomerID != src.CustomerID
    OR dest.LastName != src.LastName
    OR dest.FirstName != src.FirstName
    OR dest.MiddleInitial != src.MiddleInitial
    OR dest.GENDER != src.GENDER
    OR dest.DateOfBirth != src.DateOfBirth
    OR dest.PhonePrimary != src.PhonePrimary
    OR dest.PhoneSecondary != src.PhoneSecondary
    OR dest.EmailPrimary != src.EmailPrimary
    OR dest.EmailSecondary != src.EmailSecondary
    OR dest.AddressLine1 != src.AddressLine1
    OR dest.AddressLine2 != src.AddressLine2
    OR dest.TownName != src.TownName
    OR dest.StateProvinceCode != src.StateProvinceCode
    OR dest.PostalCode != src.PostalCode
    OR dest.CountryName != src.CountryName
    OR dest.CustomerStatusCode != src.CustomerStatusCode
    OR dest.CustomerStatusDesc != src.CustomerStatusDesc) THEN
    UPDATE SET
        dest.AccountName = src.AccountName,
        dest.TaxStatusCode = src.TaxStatusCode,
        dest.TaxStatusDesc = src.TaxStatusDesc,

```

```

dest.CustomerID = src.CustomerID,
dest.LastName = src.LastName,
dest.FirstName = src.FirstName,
dest.MiddleInitial = src.MiddleInitial,
dest.GENDER = src.GENDER,
dest.DateOfBirth = src.DateOfBirth,
dest.PhonePrimary = src.PhonePrimary,
dest.PhoneSecondary = src.PhoneSecondary,
dest.EmailPrimary = src.EmailPrimary,
dest.EmailSecondary = src.EmailSecondary,
dest.AddressLine1 = src.AddressLine1,
dest.AddressLine2 = src.AddressLine2,
dest.TownName = src.TownName,
dest.StateProvinceCode = src.StateProvinceCode,
dest.PostalCode = src.PostalCode,
dest.CountryName = src.CountryName,
dest.CustomerStatusCode = src.CustomerStatusCode,
dest.CustomerStatusDesc = src.CustomerStatusDesc;

```

```
SET @QueryEnd = SYSDATETIME();
```

```
DECLARE
```

```

    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
```

```
SELECT @CountTotal = COUNT(*) FROM D_CustomerAccount;
```

```
SELECT @CountInsert = COUNT(*) FROM D_CustomerAccount WHERE WH_StartDate = @BatchDate;
```

```
SELECT @CountUpdate = COUNT(*) FROM D_CustomerAccount WHERE WH_EndDate = DATEADD(SECOND,-1,@BatchDate);
```

```
INSERT INTO master.dbo.LoadResults
```

```
SELECT
```

```

    @Iteration,
    'DM',
    'D_CustomerAccount',
    'Dimension',
    @QueryStart,
    @QueryEnd,

```

```
@CountInsert,  
@CountUpdate,  
@CountTotal;
```

```
GO
```

```
/*  
D_Exchange  
*/
```

```
USE TPCE_DM;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);
```

```
MERGE dbo.D_Exchange AS dest  
USING dbo.V_D_Exchange AS src WITH (NOLOCK)  
    ON dest.ExchangeID = src.ExchangeID  
WHEN NOT MATCHED THEN  
    INSERT VALUES  
    (  
        src.ExchangeID,  
        src.ExchangeName,  
        src.ExchangeDesc,  
        src.OpeningTime,  
        src.ClosingTime,  
        src.ListedSecurities,  
        src.AddressLine1,  
        src.AddressLine2,  
        src.TownName,  
        src.StateProvinceCode,  
        src.PostalCode,
```

```

        src.CountryName,
        @BatchDate,
        NULL
    )
WHEN MATCHED
    AND (dest.ExchangeName != src.ExchangeName
    OR dest.ExchangeDesc != src.ExchangeDesc
    OR dest.OpeningTime != src.OpeningTime
    OR dest.ClosingTime != src.ClosingTime
    OR dest.ListedSecurities != src.ListedSecurities
    OR dest.AddressLine1 != src.AddressLine1
    OR dest.AddressLine2 != src.AddressLine2
    OR dest.TownName != src.TownName
    OR dest.StateProvinceCode != src.StateProvinceCode
    OR dest.PostalCode != src.PostalCode
    OR dest.CountryName != src.CountryName) THEN
    UPDATE SET
        dest.ExchangeName = src.ExchangeName,
        dest.ExchangeDesc = src.ExchangeDesc,
        dest.OpeningTime = src.OpeningTime,
        dest.ClosingTime = src.ClosingTime,
        dest.ListedSecurities = src.ListedSecurities,
        dest.AddressLine1 = src.AddressLine1,
        dest.AddressLine2 = src.AddressLine2,
        dest.TownName = src.TownName,
        dest.StateProvinceCode = src.StateProvinceCode,
        dest.PostalCode = src.PostalCode,
        dest.CountryName = src.CountryName,
        dest.WH_DateUpdated = @BatchDate;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM D_Exchange;

```

```
SELECT @CountInsert = COUNT(*) FROM D_Exchange WHERE WH_DateInserted = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM D_Exchange WHERE WH_DateUpdated = @BatchDate;
```

```
INSERT INTO master.dbo.LoadResults
SELECT
```

```
    @Iteration,
    'DM',
    'D_Exchange',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;
```

```
GO
```

```
/******
D_Security
******/
```

```
USE TPCE_DM;
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);
```

```
MERGE dbo.D_Security AS dest
USING dbo.V_D_Security AS src WITH (NOLOCK)
    ON dest.SecurityCode = src.SecurityCode
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
```

```

        src.SecurityCode,
        src.SecurityName,
        src.IssueType,
        src.FirstTradeDate,
        src.SecurityStatusCode,
        src.SecurityStatusDesc,
        @BatchDate,
        NULL
    )
WHEN MATCHED
    AND (dest.SecurityName != src.SecurityName
    OR dest.IssueType != src.IssueType
    OR dest.FirstTradeDate != src.FirstTradeDate
    OR dest.SecurityStatusCode != src.SecurityStatusCode
    OR dest.SecurityStatusDesc != src.SecurityStatusDesc) THEN
    UPDATE SET
        dest.SecurityName = src.SecurityName,
        dest.IssueType = src.IssueType,
        dest.FirstTradeDate = src.FirstTradeDate,
        dest.SecurityStatusCode = src.SecurityStatusCode,
        dest.SecurityStatusDesc = src.SecurityStatusDesc,
        dest.WH_DateUpdated = @BatchDate;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM D_Security;
SELECT @CountInsert = COUNT(*) FROM D_Security WHERE WH_DateInserted = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM D_Security WHERE WH_DateUpdated = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',

```

```
'D_Security',  
'Dimension',  
@QueryStart,  
@QueryEnd,  
@CountInsert,  
@CountUpdate,  
@CountTotal;
```

```
GO
```

```
/*  
D_TradeStatus  
*/
```

```
USE TPCE_DM;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();  
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);
```

```
MERGE dbo.D_TradeStatus AS dest  
USING dbo.V_D_TradeStatus AS src WITH (NOLOCK)  
    ON dest.TradeStatusCode = src.TradeStatusCode
```

```
WHEN NOT MATCHED THEN
```

```
    INSERT VALUES
```

```
    (  
        src.TradeStatusCode,  
        src.TradeStatusDesc,  
        @BatchDate,  
        NULL  
    )
```

```
WHEN MATCHED AND dest.TradeStatusDesc != src.TradeStatusDesc THEN
```

```
    UPDATE SET
```



```
dest.TradeStatusDesc = src.TradeStatusDesc,  
dest.WH_DateUpdated = @BatchDate;
```

```
SET @QueryEnd = SYSDATETIME();
```

```
DECLARE
```

```
    @Iteration AS INT,  
    @CountTotal AS INT,  
    @CountInsert AS INT,  
    @CountUpdate AS INT;
```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
```

```
SELECT @CountTotal = COUNT(*) FROM D_TradeStatus;
```

```
SELECT @CountInsert = COUNT(*) FROM D_TradeStatus WHERE WH_DateInserted = @BatchDate;
```

```
SELECT @CountUpdate = COUNT(*) FROM D_TradeStatus WHERE WH_DateUpdated = @BatchDate;
```

```
INSERT INTO master.dbo.LoadResults
```

```
SELECT
```

```
    @Iteration,  
    'DM',  
    'D_TradeStatus',  
    'Dimension',  
    @QueryStart,  
    @QueryEnd,  
    @CountInsert,  
    @CountUpdate,  
    @CountTotal;
```

```
GO
```

```
/*  
D_TradeType  
*/
```

```
USE TPCE_DM;
```

```
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),
```

```

        @QueryStart AS DATETIME2(7),
        @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.D_TradeType AS dest
USING dbo.V_D_TradeType AS src WITH (NOLOCK)
    ON dest.TradeTypeCode = src.TradeTypeCode
WHEN NOT MATCHED THEN
    INSERT VALUES
        (
            src.TradeTypeCode,
            src.TradeTypeDesc,
            src.TradeCategory,
            src.MarketTransactionFlag,
            @BatchDate,
            Null
        )
WHEN MATCHED
    AND (dest.TradeTypeDesc != src.TradeTypeDesc
    OR dest.TradeCategory != src.TradeCategory
    OR dest.MarketTransactionFlag != src.MarketTransactionFlag) THEN
    UPDATE SET
        dest.TradeTypeDesc = src.TradeTypeDesc,
        dest.TradeCategory = src.TradeCategory,
        dest.MarketTransactionFlag = src.MarketTransactionFlag,
        dest.WH_DateUpdated = @BatchDate;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM D_TradeType;

```

```

SELECT @CountInsert = COUNT(*) FROM D_TradeType WHERE WH_DateInserted = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM D_TradeType WHERE WH_DateUpdated = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'D_TradeType',
    'Dimension',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

/*****
F_DailyMarketPerformance
*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @IncrementalDate AS DATE;

SELECT @IncrementalDate = DataDate FROM master.dbo.V_IterationDates WITH (NOLOCK);
SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

ALTER INDEX IDX_F_DailyMarketPerformance_Company_SK ON TPCE_DM.dbo.F_DailyMarketPerformance DISABLE;
ALTER INDEX IDX_F_DailyMarketPerformance_Security_SK ON TPCE_DM.dbo.F_DailyMarketPerformance DISABLE;

INSERT INTO TPCE_DM.dbo.F_DailyMarketPerformance WITH (TABLOCK)

```

```

(
    Date_SK,
    Company_SK,
    Exchange_SK,
    Security_SK,
    DayClose,
    DayHigh,
    DayLow,
    DayVolume,
    [2WeekHigh],
    [2WeekLow],
    [2WeekVolume],
    WH_DateInserted
)
SELECT
    CONVERT(VARCHAR(8),dm.DM_DATE,112) AS Date_SK,
    dc.Company_SK,
    de.Exchange_SK,
    ds.Security_SK,
    dm.DM_CLOSE,
    dm.DM_HIGH,
    dm.DM_LOW,
    dm.DM_VOL,
    wk.[2WeekHigh],
    wk.[2WeekLow],
    wk.[2WeekVol],
    @BatchDate
FROM
    dbo.DAILY_MARKET dm WITH (NOLOCK)
    JOIN dbo.[SECURITY] s WITH (NOLOCK)
        ON dm.DM_S_SYMB = s.S_SYMB
    JOIN TPCE_DM.dbo.D_Security ds WITH (NOLOCK)
        ON dm.DM_S_SYMB = ds.SecurityCode
    JOIN TPCE_DM.dbo.D_Company dc WITH (NOLOCK)
        ON s.S_CO_ID = dc.CompanyID
        AND dc.WH_CurrInd = 1
    JOIN TPCE_DM.dbo.D_Exchange de WITH (NOLOCK)
        ON s.S_EX_ID = de.ExchangeID
    JOIN
        (

```

```

SELECT
    x.DM_DATE,
    x.DM_S_SYMB,
    MIN(y.DM_LOW) [2WeekLow],
    MAX(y.DM_HIGH) [2WeekHigh],
    SUM(y.DM_VOL) [2WeekVol]
FROM
    dbo.DAILY_MARKET x WITH (NOLOCK)
    JOIN dbo.DAILY_MARKET y WITH (NOLOCK)
        ON x.DM_S_SYMB = y.DM_S_SYMB
        AND y.DM_DATE >= DATEADD(WEEK, -2, x.DM_DATE)
        AND y.DM_DATE <= x.DM_DATE
WHERE
    x.DM_DATE = @IncrementalDate
GROUP BY
    x.DM_DATE,
    x.DM_S_SYMB
) wk
ON dm.DM_DATE = wk.DM_DATE
AND dm.DM_S_SYMB = wk.DM_S_SYMB
WHERE
    dm.DM_DATE = @IncrementalDate;

ALTER INDEX IDX_F_DailyMarketPerformance_Company_SK ON TPCE_DM.dbo.F_DailyMarketPerformance REBUILD;
ALTER INDEX IDX_F_DailyMarketPerformance_Security_SK ON TPCE_DM.dbo.F_DailyMarketPerformance REBUILD;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.F_DailyMarketPerformance;
SELECT @CountInsert = COUNT(*) FROM TPCE_DM.dbo.F_DailyMarketPerformance WHERE WH_DateInserted = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT

```

```

@Iteration,
'DM',
'F_DailyMarketPerformance',
'Fact',
@QueryStart,
@QueryEnd,
@CountInsert,
NULL,
@CountTotal;

```

GO

```

/*****
F_Trade
*****/

```

```

USE TPCE;
GO

```

SET NOCOUNT ON;

DECLARE

```

@BatchDate AS DATETIME2(0),
@QueryStart AS DATETIME2(7),
@QueryEnd AS DATETIME2(7),
@IncrementalDate AS INT;

```

```

SELECT @IncrementalDate = CONVERT(INT,CONVERT(VARCHAR(8),DataDate,112)) FROM master.dbo.V_IterationDates WITH (NOLOCK);
SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

```

```

ALTER INDEX IDX_F_Trade_Broker_SK ON TPCE_DM.dbo.F_Trade DISABLE;
ALTER INDEX IDX_F_Trade_Company_SK ON TPCE_DM.dbo.F_Trade DISABLE;
ALTER INDEX IDX_F_Trade_CustomerAccount_SK ON TPCE_DM.dbo.F_Trade DISABLE;
ALTER INDEX IDX_F_Trade_Date_SK ON TPCE_DM.dbo.F_Trade DISABLE;
ALTER INDEX IDX_F_Trade_Security_SK ON TPCE_DM.dbo.F_Trade DISABLE;

```

```

INSERT INTO TPCE_DM.dbo.F_Trade WITH (TABLOCK)
(
    Date_SK,

```

```
Broker_SK,  
CustomerAccount_SK,  
Company_SK,  
Exchange_SK,  
Security_SK,  
TradeType_SK,  
CashType_SK,  
TradeStatus_SK,  
TradeID,  
TradeDateTime,  
Quantity,  
BidPrice,  
TradePrice,  
Charge,  
Commission,  
Tax,  
WH_DateInserted
```

```
)
```

```
SELECT
```

```
CONVERT(VARCHAR(8),t.T_DTS,112) AS Date_SK,  
db.Broker_SK AS Broker_SK,  
dca.CustomerAccount_SK AS CustomerAccount_SK,  
dcom.Company_SK AS Company_SK,  
de.Exchange_SK AS Exchange_SK,  
ds.Security_SK AS Security_SK,  
dtt.TradeType_SK AS TradeType_SK,  
dct.CashType_SK AS CashType_SK,  
dts.TradeStatus_SK AS TradeStatus_SK,  
t.T_ID AS TradeID,  
t.T_DTS AS TradeDateTime,  
t.T_QTY AS Quantity,  
t.T_BID_PRICE AS BidPrice,  
t.T_TRADE_PRICE AS TradePrice,  
t.T_CHRG AS Charge,  
t.T_COMM AS Commission,  
t.T_TAX AS Tax,  
@BatchDate AS WH_DateInserted
```

```
FROM
```

```
dbo.TRADE t (NOLOCK)  
JOIN dbo.CUSTOMER_ACCOUNT a (NOLOCK)
```

```

        ON t.T_CA_ID = a.CA_ID
JOIN TPCE_DM.dbo.D_CustomerAccount dca (NOLOCK)
        ON a.CA_ID = dca.AccountID
        AND dca.WH_CurrInd = 1
JOIN dbo.[BROKER] b (NOLOCK)
        ON b.B_ID = a.CA_B_ID
JOIN TPCE_DM.dbo.D_Broker db (NOLOCK)
        ON db.BrokerID = b.B_ID
JOIN dbo.[SECURITY] s (NOLOCK)
        ON t.T_S_SYMB = s.S_SYMB
JOIN TPCE_DM.dbo.D_Company dcom (NOLOCK)
        ON s.S_CO_ID = dcom.CompanyID
        AND dcom.WH_CurrInd = 1
JOIN TPCE_DM.dbo.D_Exchange de (NOLOCK)
        ON s.S_EX_ID = de.ExchangeID
JOIN TPCE_DM.dbo.D_Security ds (NOLOCK)
        ON s.S_SYMB = ds.SecurityCode
JOIN TPCE_DM.dbo.D_TradeType dtt (NOLOCK)
        ON t.T_TT_ID = dtt.TradeTypeCode
JOIN TPCE_DM.dbo.D_CashType dct (NOLOCK)
        ON t.T_IS_CASH = dct.IsCashFlag
JOIN dbo.STATUS_TYPE st (NOLOCK)
        ON t.T_ST_ID = st.ST_ID
JOIN TPCE_DM.dbo.D_TradeStatus dts (NOLOCK)
        ON st.ST_ID = dts.TradeStatusCode

```

WHERE

```

    CONVERT(VARCHAR(8),t.T_DTS,112) = @IncrementalDate;

```

```

ALTER INDEX IDX_F_Trade_Broker_SK ON TPCE_DM.dbo.F_Trade REBUILD;
ALTER INDEX IDX_F_Trade_Company_SK ON TPCE_DM.dbo.F_Trade REBUILD;
ALTER INDEX IDX_F_Trade_CustomerAccount_SK ON TPCE_DM.dbo.F_Trade REBUILD;
ALTER INDEX IDX_F_Trade_Date_SK ON TPCE_DM.dbo.F_Trade REBUILD;
ALTER INDEX IDX_F_Trade_Security_SK ON TPCE_DM.dbo.F_Trade REBUILD;

```

```

SET @QueryEnd = SYSDATETIME();

```

DECLARE

```

    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,

```



```

        @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DM.dbo.F_Trade;
SELECT @CountInsert = COUNT(*) FROM TPCE_DM.dbo.F_Trade WHERE WH_DateInserted = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DM',
    'F_Trade',
    'Fact',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

/*****
DB_Size
*****/
USE TPCE_DM;
GO

dbcc updateusage(0) with no_infomsgs

DECLARE
    @pages bigint ,
    @dbsize bigint,
    @logsize bigint,
    @reservedpages bigint,
    @usedpages bigint,
    @rowCount bigint,
    @Iteration int;

select @dbsize = sum(convert(bigint,case when status & 64 = 0 then size else 0 END))
        , @logsize = sum(convert(bigint,case when status & 64 <> 0 then size else 0 END))
        from dbo.sysfiles

```

```

select @reservedpages = sum(a.total_pages),
       @usedpages = sum(a.used_pages),
       @pages = sum(
           CASE
               -- XML-Index and FT-Index internal tables are not considered "data", but is part of "index_size"
               When it.internal_type IN (202,204,211,212,213,214,215,216) Then 0
               When a.type <> 1 Then a.used_pages
               When p.index_id < 2 Then a.data_pages
               Else 0
           END
       )
from sys.partitions p join sys.allocation_units a on p.partition_id = a.container_id
left join sys.internal_tables it on p.object_id = it.object_id

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;

```

```

INSERT INTO master.dbo.SizeResults

```

```

SELECT

```

```

    @Iteration,
    'DM',
    CONVERT(BIGINT,ltrim(str(@pages * 8192 / 1024.,15,0) + '')),
    CONVERT(BIGINT,ltrim(str((@usedpages - @pages) * 8192 / 1024.,15,0) + ''));

```

### 1.2.2.3 TPCE\_DV Incremental Data Loading T-SQL Script

```

/*****
dbo.HUB_Account
*****/
USE TPCE_DV;
GO

USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE

```

```

    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.HUB_Account AS dest
USING dbo.V_HUB_Account AS src WITH (NOLOCK)
    ON dest.AccountID = src.AccountID
WHEN NOT MATCHED THEN
    INSERT VALUES
        (
            @BatchDate,
            'TPCE',
            src.AccountID
        );

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM HUB_Account;
SELECT @CountInsert = COUNT(*) FROM HUB_Account WHERE HUB_Account_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_Account',
    'Hub',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

```

```

/*****
dbo.HUB_Broker
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.HUB_Broker AS dest
USING dbo.V_HUB_Broker AS src WITH (NOLOCK)
    ON dest.BrokerID = src.BrokerID
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
        @BatchDate,
        'TPCE',
        src.BrokerID
    );

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM HUB_Broker;
SELECT @CountInsert = COUNT(*) FROM HUB_Broker WHERE HUB_Broker_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT

```

```

        @Iteration,
        'DV',
        'HUB_Broker',
        'Hub',
        @QueryStart,
        @QueryEnd,
        @CountInsert,
        NULL,
        @CountTotal;
GO

/*****
dbo.HUB_Company
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.HUB_Company AS dest
USING dbo.V_HUB_Company AS src WITH (NOLOCK)
    ON dest.CompanyName = src.CompanyName
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
        @BatchDate,
        'TPCE',
        src.CompanyName
    );

SET @QueryEnd = SYSDATETIME();

```

```

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM HUB_Company;
SELECT @CountInsert = COUNT(*) FROM HUB_Company WHERE HUB_Company_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_Company',
    'Hub',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

/*****
dbo.HUB_Customer
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.HUB_Customer AS dest
USING dbo.V_HUB_Customer AS src WITH (NOLOCK)

```

```

        ON dest.CustomerID = src.CustomerID
WHEN NOT MATCHED THEN
    INSERT VALUES
        (
            @BatchDate,
            'TPCE',
            src.CustomerID
        );

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM HUB_Customer;
SELECT @CountInsert = COUNT(*) FROM HUB_Customer WHERE HUB_Customer_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_Customer',
    'Hub',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

/*****
dbo.HUB_Exchange
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.HUB_Exchange AS dest
USING dbo.V_HUB_Exchange AS src WITH (NOLOCK)
    ON dest.ExchangeID = src.ExchangeID
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
        @BatchDate,
        'TPCE',
        src.ExchangeID
    );

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM HUB_Exchange;
SELECT @CountInsert = COUNT(*) FROM HUB_Exchange WHERE HUB_Exchange_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_Exchange',
    'Hub',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,

```



```

        @CountTotal;

GO

/*****
dbo.HUB_Security
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.HUB_Security AS dest
USING dbo.V_HUB_Security AS src WITH (NOLOCK)
    ON dest.SecurityCode = src.SecurityCode
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
        @BatchDate,
        'TPCE',
        src.SecurityCode
    );

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM HUB_Security;
SELECT @CountInsert = COUNT(*) FROM HUB_Security WHERE HUB_Security_LDTS = @BatchDate;

```

```

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_Security',
    'Hub',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

/*****
dbo.HUB_TradeType
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.HUB_TradeType AS dest
USING dbo.V_HUB_TradeType AS src WITH (NOLOCK)
    ON dest.TradeTypeDesc = src.TradeTypeDesc
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
        @BatchDate,
        'TPCE',
        src.TradeTypeDesc
    );

```

```

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM HUB_TradeType;
SELECT @CountInsert = COUNT(*) FROM HUB_TradeType WHERE HUB_TradeType_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'HUB_TradeType',
    'Hub',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

/*****
dbo.XREF_Industry
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

```

```

MERGE dbo.XREF_Industry AS dest
USING dbo.V_XREF_Industry AS src WITH (NOLOCK)
    ON dest.IndustryCode = src.IndustryCode
WHEN NOT MATCHED THEN
    INSERT VALUES
        (
            src.IndustryCode,
            @BatchDate,
            'TPCE',
            src.IndustryDesc
        )
WHEN MATCHED
    AND (dest.IndustryDesc != src.IndustryDesc) THEN
    UPDATE SET
        dest.XREF_Industry_LDTS = @Batchdate,
        dest.IndustryDesc = src.IndustryDesc;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM XREF_Industry;
SELECT @CountInsert = COUNT(*) FROM XREF_Industry WHERE XREF_Industry_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'XREF_Industry',
    'Reference',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

```

GO

```
/******  
dbo.XREF_Sector  
******/
```

USE TPCE\_DV;

GO

SET NOCOUNT ON;

DECLARE

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7);
```

SET @QueryStart = SYSDATETIME();

SELECT @BatchDate = BatchDate FROM master.dbo.V\_IterationDates WITH (NOLOCK);

MERGE dbo.XREF\_Sector AS dest

USING dbo.V\_XREF\_Sector AS src WITH (NOLOCK)

ON dest.SectorCode = SRC.SectorCode

WHEN NOT MATCHED THEN

INSERT VALUES

```
(  
    src.SectorCode,  
    @BatchDate,  
    'TPCE',  
    src.SectorDesc  
)
```

WHEN MATCHED

AND dest.SectorDesc != src.SectorDesc THEN

UPDATE SET

```
    dest.XREF_Sector_LDTS = @BatchDate,  
    dest.SectorDesc = src.SectorDesc;
```

SET @QueryEnd = SYSDATETIME();

DECLARE

```
    @Iteration AS INT,  
    @CountTotal AS INT,
```

```

        @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM XREF_Sector;
SELECT @CountInsert = COUNT(*) FROM XREF_Sector WHERE XREF_Sector_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'XREF_Sector',
    'Reference',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

/*****
dbo.XREF_Status
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.XREF_Status AS dest
USING dbo.V_XREF_Status AS src WITH (NOLOCK)
    ON dest.StatusCode = src.StatusCode
WHEN NOT MATCHED THEN
    INSERT VALUES

```

```

        (
            src.StatusCode,
            @BatchDate,
            'TPCE',
            src.StatusDesc
        )
WHEN MATCHED
    AND dest.StatusDesc != src.StatusDesc THEN
    UPDATE SET
        dest.XREF_Status_LDTS = @BatchDate,
        dest.StatusDesc = src.StatusDesc;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM XREF_Status;
SELECT @CountInsert = COUNT(*) FROM XREF_Status WHERE XREF_Status_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'XREF_Status',
    'Reference',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO
/*****
dbo.XREF_Town
*****/
USE TPCE_DV;
GO

```

```

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.XREF_Town AS dest
USING dbo.V_XREF_Town AS src WITH (NOLOCK)
    ON dest.PostalCode = src.PostalCode
WHEN NOT MATCHED THEN
    INSERT VALUES
        (
            src.PostalCode,
            @BatchDate,
            'TPCE',
            src.TownName,
            src.StateProvinceCode,
            src.CountryName
        )
WHEN MATCHED
    AND (dest.TownName != src.TownName
    OR dest.StateProvinceCode != src.StateProvinceCode
    OR dest.CountryName != src.CountryName) THEN
    UPDATE SET
        dest.XREF_Town_LDTS = @BatchDate,
        dest.TownName = src.TownName,
        dest.StateProvinceCode = src.StateProvinceCode,
        dest.CountryName = src.CountryName;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

```



```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM XREF_Town;
SELECT @CountInsert = COUNT(*) FROM XREF_Town WHERE XREF_Town_LDTS = @BatchDate;
```

```
INSERT INTO master.dbo.LoadResults
SELECT
```

```
    @Iteration,
    'DV',
    'XREF_Town',
    'Reference',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;
```

```
GO
```

```
/******
dbo.LNK_AccountBroker
******/
```

```
USE TPCE_DV;
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);
```

```
SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);
```

```
MERGE dbo.LNK_AccountBroker AS dest
USING dbo.V_LNK_AccountBroker AS src WITH (NOLOCK)
    ON dest.HUB_Account_SQN = src.HUB_Account_SQN
    AND dest.HUB_Broker_SQN = src.HUB_Broker_SQN
WHEN NOT MATCHED THEN
```

```

INSERT VALUES
(
    @BatchDate,
    'TPCE',
    src.HUB_Account_SQN,
    src.HUB_Broker_SQN
);

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM LNK_AccountBroker;
SELECT @CountInsert = COUNT(*) FROM LNK_AccountBroker WHERE LNK_AccountBroker_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'LNK_AccountBroker',
    'Link',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

/*****
dbo.LNK_AccountCustomer
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.LNK_AccountCustomer AS dest
USING dbo.V_LNK_AccountCustomer AS src WITH (NOLOCK)
    ON dest.HUB_Account_SQN = src.HUB_Account_SQN
    AND dest.HUB_Customer_SQN = src.HUB_Customer_SQN
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
        @BatchDate,
        'TPCE',
        src.HUB_Account_SQN,
        src.HUB_Customer_SQN
    );

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM LNK_AccountCustomer;
SELECT @CountInsert = COUNT(*) FROM LNK_AccountCustomer WHERE LNK_AccountCustomer_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'LNK_AccountCustomer',
    'Link',
    @QueryStart,
    @QueryEnd,

```

```

        @CountInsert,
        NULL,
        @CountTotal;
GO

/*****
dbo.LNK_SecurityCompany
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.LNK_SecurityCompany AS dest
USING dbo.V_LNK_SecurityCompany AS src WITH (NOLOCK)
    ON dest.HUB_Security_SQN = src.HUB_Security_SQN
    AND dest.HUB_Company_SQN = src.HUB_Company_SQN
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
        @BatchDate,
        'TPCE',
        src.HUB_Security_SQN,
        src.HUB_Company_SQN
    );

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM LNK_SecurityCompany;
SELECT @CountInsert = COUNT(*) FROM LNK_SecurityCompany WHERE LNK_SecurityCompany_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'LNK_SecurityCompany',
    'Link',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

/*****
dbo.LNK_SecurityExchange
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

MERGE dbo.LNK_SecurityExchange AS dest
USING dbo.V_LNK_SecurityExchange AS src WITH (NOLOCK)
    ON dest.HUB_Security_SQN = src.HUB_Security_SQN
    AND dest.HUB_Exchange_SQN = src.HUB_Exchange_SQN
WHEN NOT MATCHED THEN
    INSERT VALUES

```

```

(
    @BatchDate,
    'TPCE',
    src.HUB_Security_SQN,
    src.HUB_Exchange_SQN
);

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM LNK_SecurityExchange;
SELECT @CountInsert = COUNT(*) FROM LNK_SecurityExchange WHERE LNK_SecurityExchange_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'LNK_SecurityExchange',
    'Link',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

GO

/*****
dbo.SAT_Account
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE

```

```

@BatchDate AS DATETIME2(0),
@QueryStart AS DATETIME2(7),
@QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_Account WITH (TABLOCK)
(
    HUB_Account_SQN,
    SAT_Account_LDTS,
    SAT_Account_RSRC,
    AccountName,
    TaxStatusCode
)
SELECT
    mrg.HUB_Account_SQN,
    @BatchDate,
    'TPCE',
    mrg.AccountName,
    mrg.TaxStatusCode
FROM (
    MERGE dbo.SAT_Account AS dest
    USING dbo.V_SAT_Account AS src WITH (NOLOCK)
        ON dest.HUB_Account_SQN = src.HUB_Account_SQN
        AND dest.SAT_Account_LDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT VALUES
            (
                src.HUB_Account_SQN,
                @BatchDate,
                NULL,
                'TPCE',
                src.AccountName,
                src.TaxStatusCode
            )
    WHEN MATCHED
        AND (dest.AccountName != src.AccountName
        OR dest.TaxStatusCode != src.TaxStatusCode) THEN
        UPDATE SET

```

```

        dest.SAT_Account_LEDTS = DATEADD(SECOND, -1, @BatchDate)
    OUTPUT
        $ACTION MergeAction,
        src.HUB_Account_SQN,
        src.AccountName,
        src.TaxStatusCode
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_Account;
SELECT @CountInsert = COUNT(*) FROM SAT_Account WHERE SAT_Account_LEDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_Account WHERE SAT_Account_LEDTS = DATEADD(SECOND, -1, @BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_Account',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

/*****
dbo.SAT_Broker
*****/
USE TPCE_DV;

```



```

GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_Broker WITH (TABLOCK)
(
    HUB_Broker_SQN,
    SAT_Broker_LDTS,
    SAT_Broker_RSRC,
    BrokerName,
    StatusCode
)
SELECT
    HUB_Broker_SQN,
    @BatchDate,
    'TPCE',
    BrokerName,
    StatusCode
FROM (
    MERGE dbo.SAT_Broker AS dest
    USING dbo.V_SAT_Broker AS src WITH (NOLOCK)
        ON dest.HUB_Broker_SQN = src.HUB_Broker_SQN
        AND dest.SAT_Broker_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT VALUES
            (
                src.HUB_Broker_SQN,
                @BatchDate,
                NULL,
                'TPCE',
                src.BrokerName,
                src.StatusCode
            )

```

```

    )
    WHEN MATCHED
        AND (dest.BrokerName != src.BrokerName
            OR dest.StatusCode != src.StatusCode) THEN
            UPDATE SET
                dest.SAT_Broker_LEDTS = DATEADD(SECOND,-1,@BatchDate)
    OUTPUT
        $ACTION MergeAction,
        src.HUB_Broker_SQN,
        src.BrokerName,
        src.StatusCode
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_Broker;
SELECT @CountInsert = COUNT(*) FROM SAT_Broker WHERE SAT_Broker_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_Broker WHERE SAT_Broker_LEDTS = DATEADD(SECOND,-1,@BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_Broker',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

```

```

/*****
dbo.SAT_Company
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_Company WITH (TABLOCK)
(
    HUB_Company_SQN,
    SAT_Company_LDTS,
    SAT_Company_RSRC,
    CompanyID,
    CompanyDesc,
    CompanyOpenDate,
    AddressLine1,
    AddressLine2,
    PostalCode,
    IndustryCode,
    SectorCode,
    StatusCode
)
SELECT
    mrg.HUB_Company_SQN,
    @BatchDate,
    'TPCE',
    mrg.CompanyID,
    mrg.CompanyDesc,
    mrg.CompanyOpenDate,
    mrg.AddressLine1,

```

```

mrg.AddressLine2,
mrg.PostalCode,
mrg.IndustryCode,
mrg.SectorCode,
mrg.StatusCode
FROM (
MERGE dbo.SAT_Company AS dest
USING dbo.V_SAT_Company AS src WITH (NOLOCK)
    ON dest.HUB_Company_SQN = src.HUB_Company_SQN
    AND dest.SAT_Company_LEDTS IS NULL
WHEN NOT MATCHED THEN
    INSERT VALUES
        (
            src.HUB_Company_SQN,
            @BatchDate,
            NULL,
            'TPCE',
            src.CompanyID,
            src.CompanyDesc,
            src.CompanyOpenDate,
            src.AddressLine1,
            src.AddressLine2,
            src.PostalCode,
            src.IndustryCode,
            src.SectorCode,
            src.StatusCode
        )
WHEN MATCHED
    AND (dest.CompanyID != src.CompanyID
    OR dest.CompanyDesc != src.CompanyDesc
    OR dest.CompanyOpenDate != src.CompanyOpenDate
    OR dest.AddressLine1 != src.AddressLine1
    OR dest.AddressLine2 != src.AddressLine2
    OR dest.PostalCode != src.PostalCode
    OR dest.IndustryCode != src.IndustryCode
    OR dest.SectorCode != src.SectorCode
    OR dest.StatusCode != src.StatusCode) THEN
    UPDATE SET
        dest.SAT_Company_LEDTS = DATEADD(SECOND, -1, @BatchDate)
OUTPUT

```

```

        $ACTION MergeAction,
        src.HUB_Company_SQN,
        src.CompanyID,
        src.CompanyDesc,
        src.CompanyOpenDate,
        src.AddressLine1,
        src.AddressLine2,
        src.PostalCode,
        src.IndustryCode,
        src.SectorCode,
        src.StatusCode
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_Company;
SELECT @CountInsert = COUNT(*) FROM SAT_Company WHERE SAT_Company_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_Company WHERE SAT_Company_LEDTS = DATEADD(SECOND, -1, @BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_Company',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

```

```

/*****
dbo.SAT_CompanyCEO
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_CompanyCEO WITH (TABLOCK)
(
    HUB_Company_SQN,
    SAT_CompanyCEO_LDTS,
    SAT_CompanyCEO_RSRC,
    CompanyCEO
)
SELECT
    mrg.HUB_Company_SQN,
    @BatchDate,
    'TPCE',
    mrg.CompanyCEO
FROM (
    MERGE INTO dbo.SAT_CompanyCEO AS dest
    USING dbo.V_SAT_CompanyCEO AS src WITH (NOLOCK)
        ON dest.HUB_Company_SQN = src.HUB_Company_SQN
        AND dest.SAT_CompanyCEO_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT VALUES
        (
            src.HUB_Company_SQN,
            @BatchDate,
            NULL,

```

```

        'TPCE',
        src.CompanyCEO
    )
WHEN MATCHED
    AND dest.CompanyCEO != src.CompanyCEO THEN
    UPDATE SET
        dest.SAT_CompanyCEO_LEDTS = DATEADD(SECOND,-1,@BatchDate)

OUTPUT
    $ACTION MergeAction,
    src.HUB_Company_SQN,
    src.CompanyCEO
) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_CompanyCEO;
SELECT @CountInsert = COUNT(*) FROM SAT_CompanyCEO WHERE SAT_CompanyCEO_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_CompanyCEO WHERE SAT_CompanyCEO_LEDTS = DATEADD(SECOND,-1,@BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_CompanyCEO',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

```

```

/*****
dbo.SAT_CompanyCreditRating
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_CompanyCreditRating WITH (TABLOCK)
(
    HUB_Company_SQN,
    SAT_CompanyCreditRating_LDTS,
    SAT_CompanyCreditRating_RSRC,
    CompanyCreditRating
)
SELECT
    mrg.HUB_Company_SQN,
    @BatchDate,
    'TPCE',
    mrg.CompanyCreditRating
FROM (
    MERGE dbo.SAT_CompanyCreditRating AS dest
    USING dbo.V_SAT_CompanyCreditRating AS src WITH (NOLOCK)
        ON dest.HUB_Company_SQN = src.HUB_Company_SQN
        AND dest.SAT_CompanyCreditRating_LDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT VALUES
        (
            src.HUB_Company_SQN,
            @BatchDate,
            NULL,

```



```

        'TPCE',
        src.CompanyCreditRating
    )
WHEN MATCHED
    AND dest.CompanyCreditRating != src.CompanyCreditRating THEN
    UPDATE SET
        dest.SAT_CompanyCreditRating_LEDTS = DATEADD(SECOND,-1,@BatchDate)

OUTPUT
    $ACTION MergeAction,
    src.HUB_Company_SQN,
    src.CompanyCreditRating
) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_CompanyCreditRating;
SELECT @CountInsert = COUNT(*) FROM SAT_CompanyCreditRating WHERE SAT_CompanyCreditRating_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_CompanyCreditRating WHERE SAT_CompanyCreditRating_LEDTS = DATEADD(SECOND,-1,@BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_CompanyCreditRating',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

```

```

/*****
dbo.SAT_Customer
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_Customer WITH (TABLOCK)
(
    HUB_Customer_SQN,
    SAT_Customer_LDTS,
    SAT_Customer_RSRC,
    LastName,
    FirstName,
    MiddleInitial,
    GENDER,
    DateOfBirth,
    TierNumber,
    StatusCode
)
SELECT
    mrg.HUB_Customer_SQN,
    @BatchDate,
    'TPCE',
    mrg.LastName,
    mrg.FirstName,
    mrg.MiddleInitial,
    mrg.GENDER,
    mrg.DateOfBirth,
    mrg.TierNumber,

```

```

    mrg.StatusCode
FROM (
    MERGE dbo.SAT_Customer AS dest
    USING dbo.V_SAT_Customer AS src WITH (NOLOCK)
        ON dest.HUB_Customer_SQN = src.HUB_Customer_SQN
        AND dest.SAT_Customer_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT VALUES
            (
                src.HUB_Customer_SQN,
                @BatchDate,
                NULL,
                'TPCE',
                src.LastName,
                src.FirstName,
                src.MiddleInitial,
                src.GENDER,
                src.DateOfBirth,
                src.TierNumber,
                src.StatusCode
            )
    WHEN MATCHED
        AND (dest.LastName != src.LastName
        OR dest.FirstName != src.FirstName
        OR dest.MiddleInitial != src.MiddleInitial
        OR dest.GENDER != src.GENDER
        OR dest.DateOfBirth != src.DateOfBirth
        OR dest.TierNumber != src.TierNumber
        OR dest.StatusCode != src.StatusCode) THEN
        UPDATE SET
            dest.SAT_Customer_LEDTS = DATEADD(SECOND,-1,@BatchDate)
    OUTPUT
        $ACTION MergeAction,
        src.HUB_Customer_SQN,
        src.LastName,
        src.FirstName,
        src.MiddleInitial,
        src.GENDER,
        src.DateOfBirth,
        src.TierNumber,

```

```

        src.StatusCode
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_Customer;
SELECT @CountInsert = COUNT(*) FROM SAT_Customer WHERE SAT_Customer_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_Customer WHERE SAT_Customer_LEDTS = DATEADD(SECOND,-1,@BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_Customer',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

/*****
dbo.SAT_CustomerContact
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE

```

```

    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_CustomerContact WITH (TABLOCK)
(
    HUB_Customer_SQN,
    SAT_CustomerContact_LDTS,
    SAT_CustomerContact_RSRC,
    PhoneCountryCodePrimary,
    PhoneAreaCodePrimary,
    PhoneLocalNumberPrimary,
    PhoneExtensionPrimary,
    PhoneCountryCodeSecondary,
    PhoneAreaCodeSecondary,
    PhoneLocalNumberSecondary,
    PhoneExtensionSecondary,
    EmailPrimary,
    EmailSecondary,
    AddressLine1,
    AddressLine2,
    PostalCode
)
SELECT
    mrg.HUB_Customer_SQN,
    @BatchDate,
    'TPCE',
    mrg.PhoneCountryCodePrimary,
    mrg.PhoneAreaCodePrimary,
    mrg.PhoneLocalNumberPrimary,
    mrg.PhoneExtensionPrimary,
    mrg.PhoneCountryCodeSecondary,
    mrg.PhoneAreaCodeSecondary,
    mrg.PhoneLocalNumberSecondary,
    mrg.PhoneExtensionSecondary,
    mrg.EmailPrimary,
    mrg.EmailSecondary,

```

```

mrg.AddressLine1,
mrg.AddressLine2,
mrg.PostalCode
FROM (
    MERGE dbo.SAT_CustomerContact AS dest
    USING dbo.V_SAT_CustomerContact AS src WITH (NOLOCK)
        ON dest.HUB_Customer_SQN = src.HUB_Customer_SQN
        AND dest.SAT_CustomerContact_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT VALUES
            (
                src.HUB_Customer_SQN,
                @BatchDate,
                NULL,
                'TPCE',
                src.PhoneCountryCodePrimary,
                src.PhoneAreaCodePrimary,
                src.PhoneLocalNumberPrimary,
                src.PhoneExtensionPrimary,
                src.PhoneCountryCodeSecondary,
                src.PhoneAreaCodeSecondary,
                src.PhoneLocalNumberSecondary,
                src.PhoneExtensionSecondary,
                src.EmailPrimary,
                src.EmailSecondary,
                src.AddressLine1,
                src.AddressLine2,
                src.PostalCode
            )
    WHEN MATCHED
        AND (dest.PhoneCountryCodePrimary != src.PhoneCountryCodePrimary
        OR dest.PhoneAreaCodePrimary != src.PhoneAreaCodePrimary
        OR dest.PhoneLocalNumberPrimary != src.PhoneLocalNumberPrimary
        OR dest.PhoneExtensionPrimary != src.PhoneExtensionPrimary
        OR dest.PhoneCountryCodeSecondary != src.PhoneCountryCodeSecondary
        OR dest.PhoneAreaCodeSecondary != src.PhoneAreaCodeSecondary
        OR dest.PhoneLocalNumberSecondary != src.PhoneLocalNumberSecondary
        OR dest.PhoneExtensionSecondary != src.PhoneExtensionSecondary
        OR dest.EmailPrimary != src.EmailPrimary
        OR dest.EmailSecondary != src.EmailSecondary

```

```
OR dest.AddressLine1 != src.AddressLine1
OR dest.AddressLine2 != src.AddressLine2
OR dest.PostalCode != src.PostalCode) THEN
```

```
UPDATE SET
```

```
dest.SAT_CustomerContact_LEDTS = DATEADD(SECOND, -1, @BatchDate)
```

```
OUTPUT
```

```
$ACTION MergeAction,
src.HUB_Customer_SQN,
src.PhoneCountryCodePrimary,
src.PhoneAreaCodePrimary,
src.PhoneLocalNumberPrimary,
src.PhoneExtensionPrimary,
src.PhoneCountryCodeSecondary,
src.PhoneAreaCodeSecondary,
src.PhoneLocalNumberSecondary,
src.PhoneExtensionSecondary,
src.EmailPrimary,
src.EmailSecondary,
src.AddressLine1,
src.AddressLine2,
src.PostalCode
```

```
) AS mrg
```

```
WHERE
```

```
mrg.MergeAction = 'UPDATE';
```

```
SET @QueryEnd = SYSDATETIME();
```

```
DECLARE
```

```
@Iteration AS INT,
@CountTotal AS INT,
@CountInsert AS INT,
@CountUpdate AS INT;
```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
```

```
SELECT @CountTotal = COUNT(*) FROM SAT_CustomerContact;
```

```
SELECT @CountInsert = COUNT(*) FROM SAT_CustomerContact WHERE SAT_CustomerContact_LDTS = @BatchDate;
```

```
SELECT @CountUpdate = COUNT(*) FROM SAT_CustomerContact WHERE SAT_CustomerContact_LEDTS = DATEADD(SECOND, -1, @BatchDate);
```

```
INSERT INTO master.dbo.LoadResults
```

```
SELECT
```

```

        @Iteration,
        'DV',
        'SAT_CustomerContact',
        'Satellite',
        @QueryStart,
        @QueryEnd,
        @CountInsert,
        @CountUpdate,
        @CountTotal;
GO

/*****
dbo.SAT_Exchange
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_Exchange WITH (TABLOCK)
(
    HUB_Exchange_SQN,
    SAT_Exchange_LDTS,
    SAT_Exchange_RSRC,
    ExchangeName,
    ExchangeDesc,
    OpeningTime,
    ClosingTime,
    AddressLine1,
    AddressLine2,
    PostalCode
)

```



```

SELECT
    mrg.HUB_Exchange_SQN,
    @BatchDate,
    'TPCE',
    mrg.ExchangeName,
    mrg.ExchangeDesc,
    mrg.OpeningTime,
    mrg.ClosingTime,
    mrg.AddressLine1,
    mrg.AddressLine2,
    mrg.PostalCode
FROM
    (
        MERGE dbo.SAT_Exchange AS dest
        USING dbo.V_SAT_Exchange AS src WITH (NOLOCK)
            ON dest.HUB_Exchange_SQN = src.HUB_Exchange_SQN
            AND dest.SAT_Exchange_LEDTS IS NULL
        WHEN NOT MATCHED THEN
            INSERT VALUES
                (
                    src.HUB_Exchange_SQN,
                    @BatchDate,
                    NULL,
                    'TPCE',
                    src.ExchangeName,
                    src.ExchangeDesc,
                    src.OpeningTime,
                    src.ClosingTime,
                    src.AddressLine1,
                    src.AddressLine2,
                    src.PostalCode
                )
        WHEN MATCHED
            AND (dest.ExchangeName != src.ExchangeName
            OR dest.ExchangeDesc != src.ExchangeDesc
            OR dest.OpeningTime != src.OpeningTime
            OR dest.ClosingTime != src.ClosingTime
            OR dest.AddressLine1 != src.AddressLine1
            OR dest.AddressLine2 != src.AddressLine2
            OR dest.PostalCode != src.PostalCode) THEN

```

```

        UPDATE SET
            dest.SAT_Exchange_LEDTS = DATEADD(SECOND,-1,@BatchDate)
    OUTPUT
        $ACTION MergeAction,
        src.HUB_Exchange_SQN,
        src.ExchangeName,
        src.ExchangeDesc,
        src.OpeningTime,
        src.ClosingTime,
        src.AddressLine1,
        src.AddressLine2,
        src.PostalCode
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_Exchange;
SELECT @CountInsert = COUNT(*) FROM SAT_Exchange WHERE SAT_Exchange_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_Exchange WHERE SAT_Exchange_LEDTS = DATEADD(SECOND,-1,@BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_Exchange',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

```

```

GO

/*****
dbo.SAT_ExchangeSecurities
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_ExchangeSecurities WITH (TABLOCK)
(
    HUB_Exchange_SQN,
    SAT_ExchangeSecurities_LDTS,
    SAT_ExchangeSecurities_RSRC,
    ListedSecurities
)
SELECT
    mrg.HUB_Exchange_SQN,
    @BatchDate,
    'TPCE',
    mrg.ListedSecurities
FROM
(
    MERGE dbo.SAT_ExchangeSecurities AS dest
    USING dbo.V_SAT_ExchangeSecurities AS src WITH (NOLOCK)
        ON dest.HUB_Exchange_SQN = src.HUB_Exchange_SQN
        AND dest.SAT_ExchangeSecurities_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT VALUES
        (
            src.HUB_Exchange_SQN,

```

```

        @BatchDate,
        NULL,
        'TPCE',
        src.ListedSecurities
    )
WHEN MATCHED
    AND dest.ListedSecurities != src.ListedSecurities THEN
    UPDATE SET
        dest.SAT_ExchangeSecurities_LEDTS = DATEADD(SECOND,-1,@BatchDate)
OUTPUT
    $ACTION MergeAction,
    src.HUB_Exchange_SQN,
    src.ListedSecurities
) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_ExchangeSecurities;
SELECT @CountInsert = COUNT(*) FROM SAT_ExchangeSecurities WHERE SAT_ExchangeSecurities_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_ExchangeSecurities WHERE SAT_ExchangeSecurities_LEDTS = DATEADD(SECOND,-1,@BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_ExchangeSecurities',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,

```

```

        @CountTotal;

GO

/*****
dbo.SAT_Security
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_Security WITH (TABLOCK)
(
    HUB_Security_SQN,
    SAT_Security_LDTS,
    SAT_Security_RSRC,
    SecurityName,
    IssueType,
    FirstTradeDate,
    StatusCode
)
SELECT
    mrg.HUB_Security_SQN,
    @BatchDate,
    'TPCE',
    mrg.SecurityName,
    mrg.IssueType,
    mrg.FirstTradeDate,
    mrg.StatusCode
FROM
    (
        MERGE INTO dbo.SAT_Security AS dest

```

```

USING dbo.V_SAT_Security AS src WITH (NOLOCK)
    ON dest.HUB_Security_SQN = src.HUB_Security_SQN
    AND dest.SAT_Security_LEDTS IS NULL
WHEN NOT MATCHED THEN
    INSERT VALUES
    (
        src.HUB_Security_SQN,
        @BatchDate,
        NULL,
        'TPCE',
        src.SecurityName,
        src.IssueType,
        src.FirstTradeDate,
        src.StatusCode
    )
WHEN MATCHED
    AND (dest.SecurityName != src.SecurityName
    OR dest.IssueType != src.IssueType
    OR dest.FirstTradeDate != src.FirstTradeDate
    OR dest.StatusCode != src.StatusCode) THEN
    UPDATE SET
        dest.SAT_Security_LEDTS = DATEADD(SECOND, -1, @BatchDate)
OUTPUT
    $ACTION MergeAction,
    src.HUB_Security_SQN,
    src.SecurityName,
    src.IssueType,
    src.FirstTradeDate,
    src.StatusCode
) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_Security;
SELECT @CountInsert = COUNT(*) FROM SAT_Security WHERE SAT_Security_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_Security WHERE SAT_Security_LEDTS = DATEADD(SECOND,-1,@BatchDate);

```

```

INSERT INTO master.dbo.LoadResults
SELECT

```

```

    @Iteration,
    'DV',
    'SAT_Security',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

```

```
GO
```

```

/*****
dbo.SAT_TradeType
*****/

```

```

USE TPCE_DV;
GO

```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```

    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

```

```

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

```

```

INSERT INTO dbo.SAT_TradeType WITH (TABLOCK)
(
    HUB_TradeType_SQN,
    SAT_TradeType_LDTS,

```

```

        SAT_TradeType_RSRC,
        TradeTypeCode,
        IsSellFlag,
        IsMarketFlag
    )
SELECT
    HUB_TradeType_SQN,
    @BatchDate,
    'TPCE',
    TradeTypeCode,
    IsSellFlag,
    IsMarketFlag
FROM (
    MERGE dbo.SAT_TradeType AS dest
    USING dbo.V_SAT_TradeType AS src WITH (NOLOCK)
        ON dest.HUB_TradeType_SQN = src.HUB_TradeType_SQN
        AND dest.SAT_TradeType_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT VALUES
            (
                src.HUB_TradeType_SQN,
                @BatchDate,
                NULL,
                'TPCE',
                src.TradeTypeCode,
                src.IsSellFlag,
                src.IsMarketFlag
            )
    WHEN MATCHED
        AND (dest.TradeTypeCode != src.TradeTypeCode
        OR dest.IsSellFlag != src.IsSellFlag
        OR dest.IsMarketFlag != src.IsMarketFlag) THEN
        UPDATE SET
            dest.SAT_TradeType_LEDTS = DATEADD(SECOND,-1,@BatchDate)
    OUTPUT
        $ACTION MergeAction,
        src.HUB_TradeType_SQN,
        src.TradeTypeCode,
        src.IsSellFlag,
        src.IsMarketFlag

```



```

        ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_TradeType;
SELECT @CountInsert = COUNT(*) FROM SAT_TradeType WHERE SAT_TradeType_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_TradeType WHERE SAT_TradeType_LEDTS = DATEADD(SECOND, -1, @BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_TradeType',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

/*****
dbo.SAT_AccountBrokerStatus
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),

```

```

        @QueryStart AS DATETIME2(7),
        @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_AccountBrokerStatus
(
    LNK_AccountBroker_SQN,
    SAT_AccountBrokerStatus_LDTS,
    SAT_AccountBrokerStatus_RSRC
)
SELECT
    mrg.LNK_AccountBroker_SQN,
    @BatchDate,
    'TPCE'
FROM (
    MERGE dbo.V_SAT_AccountBrokerStatusDetail AS dest
    USING dbo.V_SAT_AccountBrokerStatus AS src
        ON dest.AccountID = src.CA_ID
        AND SAT_AccountBrokerStatus_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT
        (
            LNK_AccountBroker_SQN,
            SAT_AccountBrokerStatus_LDTS,
            SAT_AccountBrokerStatus_RSRC
        )
        VALUES
        (
            src.LNK_AccountBroker_SQN,
            @BatchDate,
            'TPCE'
        )
    WHEN MATCHED
        AND dest.BrokerID != CA_B_ID THEN
        UPDATE SET
            dest.SAT_AccountBrokerStatus_LEDTS = DATEADD(SECOND,-1,@BatchDate)
    OUTPUT
        $Action MergeAction,

```

```

        src.LNK_AccountBroker_SQN
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_AccountBrokerStatus;
SELECT @CountInsert = COUNT(*) FROM SAT_AccountBrokerStatus WHERE SAT_AccountBrokerStatus_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_AccountBrokerStatus WHERE SAT_AccountBrokerStatus_LEDTS = DATEADD(SECOND,-1,@BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_AccountBrokerStatus',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

/*****
dbo.SAT_AccountCustomerStatus
*****/
USE TPCE_DV;
GO

SET NOCOUNT ON;

```

```

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_AccountCustomerStatus
(
    LNK_AccountCustomer_SQN,
    SAT_AccountCustomerStatus_LDTS,
    SAT_AccountCustomerStatus_RSRC
)
SELECT
    mrg.LNK_AccountCustomer_SQN,
    @BatchDate,
    'TPCE'
FROM (
    MERGE dbo.V_SAT_AccountCustomerStatusDetail AS dest
    USING dbo.V_SAT_AccountCustomerStatus AS src
        ON dest.AccountID = src.CA_ID
        AND SAT_AccountCustomerStatus_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT
        (
            LNK_AccountCustomer_SQN,
            SAT_AccountCustomerStatus_LDTS,
            SAT_AccountCustomerStatus_RSRC
        )
        VALUES
        (
            src.LNK_AccountCustomer_SQN,
            @BatchDate,
            'TPCE'
        )
    WHEN MATCHED
        AND dest.CustomerID != CA_C_ID THEN
        UPDATE SET
            dest.SAT_AccountCustomerStatus_LEDTS = DATEADD(SECOND, -1, @BatchDate)

```

```

        OUTPUT
            $Action MergeAction,
            src.LNK_AccountCustomer_SQN
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_AccountCustomerStatus;
SELECT @CountInsert = COUNT(*) FROM SAT_AccountCustomerStatus WHERE SAT_AccountCustomerStatus_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_AccountCustomerStatus WHERE SAT_AccountCustomerStatus_LEDTS = DATEADD(SECOND,-1,@BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_AccountCustomerStatus',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

/*****
dbo.SAT_SecurityCompanyStatus
*****/
USE TPCE_DV;
GO

```

```

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_SecurityCompanyStatus
(
    LNK_SecurityCompany_SQN,
    SAT_SecurityCompanyStatus_LDTS,
    SAT_SecurityCompanyStatus_RSRC
)
SELECT
    mrg.LNK_SecurityCompany_SQN,
    @BatchDate,
    'TPCE'
FROM (
    MERGE dbo.V_SAT_SecurityCompanyStatusDetail AS dest
    USING dbo.V_SAT_SecurityCompanyStatus AS src
        ON dest.SecurityCode = src.SecurityCode
        AND SAT_SecurityCompanyStatus_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT
            (
                LNK_SecurityCompany_SQN,
                SAT_SecurityCompanyStatus_LDTS,
                SAT_SecurityCompanyStatus_RSRC
            )
        VALUES
            (
                src.LNK_SecurityCompany_SQN,
                @BatchDate,
                'TPCE'
            )
    WHEN MATCHED
        AND dest.CompanyName != src.CompanyName THEN

```

```

        UPDATE SET
            dest.SAT_SecurityCompanyStatus_LEDTS = DATEADD(SECOND, -1, @BatchDate)
    OUTPUT
        $Action MergeAction,
        src.LNK_SecurityCompany_SQN
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_SecurityCompanyStatus;
SELECT @CountInsert = COUNT(*) FROM SAT_SecurityCompanyStatus WHERE SAT_SecurityCompanyStatus_LDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_SecurityCompanyStatus WHERE SAT_SecurityCompanyStatus_LEDTS = DATEADD(SECOND, -1, @BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_SecurityCompanyStatus',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

/*****
dbo.SAT_SecurityExchangeStatus
*****/
USE TPCE_DV;

```

```

GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7);

SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

INSERT INTO dbo.SAT_SecurityExchangeStatus
(
    LNK_SecurityExchange_SQN,
    SAT_SecurityExchangeStatus_LDTS,
    SAT_SecurityExchangeStatus_RSRC
)
SELECT
    mrg.LNK_SecurityExchange_SQN,
    @BatchDate,
    'TPCE'
FROM (
    MERGE dbo.V_SAT_SecurityExchangeStatusDetail AS dest
    USING dbo.V_SAT_SecurityExchangeStatus AS src
        ON dest.SecurityCode = src.SecurityCode
        AND SAT_SecurityExchangeStatus_LEDTS IS NULL
    WHEN NOT MATCHED THEN
        INSERT
        (
            LNK_SecurityExchange_SQN,
            SAT_SecurityExchangeStatus_LDTS,
            SAT_SecurityExchangeStatus_RSRC
        )
        VALUES
        (
            src.LNK_SecurityExchange_SQN,
            @BatchDate,
            'TPCE'
        )
)

```



```

        WHEN MATCHED
            AND dest.ExchangeID != src.ExchangeID THEN
                UPDATE SET
                    dest.SAT_SecurityExchangeStatus_LEDTS = DATEADD(SECOND,-1,@BatchDate)

    OUTPUT
        $Action MergeAction,
        src.LNK_SecurityExchange_SQN
    ) AS mrg
WHERE
    mrg.MergeAction = 'UPDATE';

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT,
    @CountUpdate AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM SAT_SecurityExchangeStatus;
SELECT @CountInsert = COUNT(*) FROM SAT_SecurityExchangeStatus WHERE SAT_SecurityExchangeStatus_LEDTS = @BatchDate;
SELECT @CountUpdate = COUNT(*) FROM SAT_SecurityExchangeStatus WHERE SAT_SecurityExchangeStatus_LEDTS = DATEADD(SECOND,-1,@BatchDate);

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'SAT_SecurityExchangeStatus',
    'Satellite',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    @CountUpdate,
    @CountTotal;

GO

/*****
dbo.TLNK_DailyMarketPerformance

```

```

*****/
USE TPCE;
GO

SET NOCOUNT ON;

DECLARE
    @BatchDate AS DATETIME2(0),
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @IncrementalDate AS DATE;

SELECT @IncrementalDate = DataDate FROM master.dbo.V_IterationDates WITH (NOLOCK);
SET @QueryStart = SYSDATETIME();
SELECT @BatchDate = BatchDate FROM master.dbo.V_IterationDates WITH (NOLOCK);

ALTER INDEX IDX_TLNK_DailyMarketPerformance_HUB_Company_SQN ON TPCE_DV.dbo.TLNK_DailyMarketPerformance DISABLE;

INSERT INTO TPCE_DV.dbo.TLNK_DailyMarketPerformance WITH (TABLOCK)
(
    HUB_Security_SQN,
    [Date],
    TLNK_DailyMarketPerformance_LDTS,
    TLNK_DailyMarketPerformance_RSRC,
    HUB_Company_SQN,
    HUB_Exchange_SQN,
    DayClose,
    DayHigh,
    DayLow,
    DayVolume
)
SELECT
    hs.HUB_Security_SQN AS HUB_Security_SQN,
    dm.DM_DATE AS [Date],
    @BatchDate AS TLNK_DailyMarketPerformance_LDTS,
    'TPCE' AS TLNK_DailyMarketPerformance_RSRC,
    hco.HUB_Company_SQN AS HUB_Company_SQN,
    hex.HUB_Exchange_SQN AS HUB_Exchange_SQN,
    dm.DM_CLOSE AS DayClose,
    dm.DM_HIGH AS DayHigh,

```

```

        dm.DM_LOW AS DayLow,
        dm.DM_VOL AS DayVolume
FROM
    dbo.DAILY_MARKET dm WITH (NOLOCK)
    JOIN TPCE_DV.dbo.HUB_Security hs WITH (NOLOCK)
        ON hs.SecurityCode = dm.DM_S_SYMB
    JOIN dbo.[SECURITY] s WITH (NOLOCK)
        ON s.S_SYMB = dm.DM_S_SYMB
    JOIN dbo.COMPANY co WITH (NOLOCK)
        ON co.CO_ID = s.S_CO_ID
    JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
        ON hco.CompanyName = co.CO_NAME
    JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)
        ON hex.ExchangeID = s.S_EX_ID
WHERE
    dm.DM_DATE = @IncrementalDate;

ALTER INDEX IDX_TLNK_DailyMarketPerformance_HUB_Company_SQN ON TPCE_DV.dbo.TLNK_DailyMarketPerformance REBUILD;
SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.TLNK_DailyMarketPerformance;
SELECT @CountInsert = COUNT(*) FROM TPCE_DV.dbo.TLNK_DailyMarketPerformance WHERE TLNK_DailyMarketPerformance_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'TLNK_DailyMarketPerformance',
    'Transactional Link',
    @QueryStart,
    @QueryEnd,
    @CountInsert,
    NULL,
    @CountTotal;

```

GO

```
/*  
*****  
dbo.TLNK_Trade  
*****  
*/
```

USE TPCE;

GO

SET NOCOUNT ON;

DECLARE

```
    @BatchDate AS DATETIME2(0),  
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7),  
    @IncrementalDate AS INT;
```

SELECT @IncrementalDate = CONVERT(INT,CONVERT(VARCHAR(8),DataDate,112)) FROM master.dbo.V\_IterationDates WITH (NOLOCK);

SET @QueryStart = SYSDATETIME();

SELECT @BatchDate = BatchDate FROM master.dbo.V\_IterationDates WITH (NOLOCK);

ALTER INDEX IDX\_TLNK\_Trade\_HUB\_Account\_SQN ON TPCE\_DV.dbo.TLNK\_Trade DISABLE;

ALTER INDEX IDX\_TLNK\_Trade\_HUB\_Broker\_SQN ON TPCE\_DV.dbo.TLNK\_Trade DISABLE;

ALTER INDEX IDX\_TLNK\_Trade\_HUB\_Company\_SQN ON TPCE\_DV.dbo.TLNK\_Trade DISABLE;

ALTER INDEX IDX\_TLNK\_Trade\_HUB\_Customer\_SQN ON TPCE\_DV.dbo.TLNK\_Trade DISABLE;

ALTER INDEX IDX\_TLNK\_Trade\_HUB\_Security\_SQN ON TPCE\_DV.dbo.TLNK\_Trade DISABLE;

INSERT INTO TPCE\_DV.dbo.TLNK\_Trade WITH (TABLOCK)

```
(  
    TradeID,  
    TLNK_Trade_LDTS,  
    TLNK_Trade_RSRC,  
    HUB_Broker_SQN,  
    HUB_Customer_SQN,  
    HUB_Account_SQN,  
    HUB_Company_SQN,  
    HUB_Exchange_SQN,  
    HUB_Security_SQN,  
    HUB_TradeType_SQN,  
    TradeDateTime,  
    Quantity,
```

```

        BidPrice,
        TradePrice,
        Charge,
        Commission,
        Tax,
        IsCashFlag,
        StatusCode
    )
SELECT
    t.T_ID AS TradeID,
    @BatchDate AS TLNK_Trade_LDTS,
    'TPCE' AS TLNK_Trade_RSRC,
    hb.HUB_Broker_SQN AS HUB_Broker_SQN,
    hc.HUB_Customer_SQN AS HUB_Customer_SQN,
    ha.HUB_Account_SQN AS HUB_Account_SQN,
    hco.HUB_Company_SQN AS HUB_Company_SQN,
    hex.HUB_Exchange_SQN AS HUB_Exchange_SQN,
    hs.HUB_Security_SQN AS HUB_Security_SQN,
    htt.HUB_TradeType_SQN AS HUB_TradeType_SQN,
    t.T_DTS AS TradeDateTime,
    t.T_QTY AS Quantity,
    t.T_BID_PRICE AS BidPrice,
    t.T_TRADE_PRICE AS TradePrice,
    t.T_CHRG AS Charge,
    t.T_COMM AS Commission,
    t.T_TAX AS Tax,
    t.T_IS_CASH AS IsCashFlag,
    t.T_ST_ID AS StatusCode
FROM
    dbo.TRADE t WITH (NOLOCK)
    JOIN dbo.CUSTOMER_ACCOUNT ca WITH (NOLOCK)
        ON ca.CA_ID = t.T_CA_ID
    JOIN TPCE_DV.dbo.HUB_Broker hb WITH (NOLOCK)
        ON hb.BrokerID = ca.CA_B_ID
    JOIN TPCE_DV.dbo.HUB_Customer hc WITH (NOLOCK)
        ON hc.CustomerID = ca.CA_C_ID
    JOIN TPCE_DV.dbo.HUB_Account ha WITH (NOLOCK)
        ON ha.AccountID = ca.CA_ID
    JOIN dbo.[SECURITY] s WITH (NOLOCK)
        ON s.S_SYMB = t.T_S_SYMB

```

```

JOIN dbo.COMPANY co WITH (NOLOCK)
    ON co.CO_ID = s.S_CO_ID
JOIN TPCE_DV.dbo.HUB_Company hco WITH (NOLOCK)
    ON hco.CompanyName = co.CO_NAME
JOIN TPCE_DV.dbo.HUB_Exchange hex WITH (NOLOCK)
    ON hex.ExchangeID = s.S_EX_ID
JOIN TPCE_DV.dbo.HUB_Security hs WITH (NOLOCK)
    ON hs.SecurityCode = s.S_SYMB
JOIN dbo.TRADE_TYPE tt WITH (NOLOCK)
    ON tt.TT_ID = t.T_TT_ID
JOIN TPCE_DV.dbo.HUB_TradeType htt WITH (NOLOCK)
    ON htt.TradeTypeDesc = tt.TT_NAME
WHERE
    CONVERT(VARCHAR(8),t.T_DTS,112) = @IncrementalDate;

ALTER INDEX IDX_TLNK_Trade_HUB_Account_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;
ALTER INDEX IDX_TLNK_Trade_HUB_Broker_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;
ALTER INDEX IDX_TLNK_Trade_HUB_Company_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;
ALTER INDEX IDX_TLNK_Trade_HUB_Customer_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;
ALTER INDEX IDX_TLNK_Trade_HUB_Security_SQN ON TPCE_DV.dbo.TLNK_Trade REBUILD;

SET @QueryEnd = SYSDATETIME();

DECLARE
    @Iteration AS INT,
    @CountTotal AS INT,
    @CountInsert AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SELECT @CountTotal = COUNT(*) FROM TPCE_DV.dbo.TLNK_Trade;
SELECT @CountInsert = COUNT(*) FROM TPCE_DV.dbo.TLNK_Trade WHERE TLNK_Trade_LDTS = @BatchDate;

INSERT INTO master.dbo.LoadResults
SELECT
    @Iteration,
    'DV',
    'TLNK_Trade',
    'Transactional Link',
    @QueryStart,
    @QueryEnd,

```

```

        @CountInsert,
        NULL,
        @CountTotal;
GO

/*****
DB_Size
*****/
USE TPCE_DV;
GO

dbcc updateusage(0) with no_infomsgs

DECLARE
    @pages bigint ,
    @dbsize bigint,
    @logsize bigint,
    @reservedpages bigint,
    @usedpages bigint,
    @rowCount bigint,
    @Iteration int;

select @dbsize = sum(convert(bigint,case when status & 64 = 0 then size else 0 END))
        , @logsize = sum(convert(bigint,case when status & 64 <> 0 then size else 0 END))
        from dbo.sysfiles

select @reservedpages = sum(a.total_pages),
       @usedpages = sum(a.used_pages),
       @pages = sum(
           CASE
               -- XML-Index and FT-Index internal tables are not considered "data", but is part of "index_size"
               When it.internal_type IN (202,204,211,212,213,214,215,216) Then 0
               When a.type <> 1 Then a.used_pages
               When p.index_id < 2 Then a.data_pages
               Else 0
           END
       )
from sys.partitions p join sys.allocation_units a on p.partition_id = a.container_id
    left join sys.internal_tables it on p.object_id = it.object_id

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;

INSERT INTO master.dbo.SizeResults
SELECT
    @Iteration,
    'DV',
    CONVERT(BIGINT,ltrim(str(@pages * 8192 / 1024.,15,0) + '')),
    CONVERT(BIGINT,ltrim(str((@usedpages - @pages) * 8192 / 1024.,15,0) + ''));

```

### 1.3 Data Balancing Scripts

```

/*****
Broker
*****/
USE TPCE_DV;
GO

SELECT *
FROM
    (
        SELECT
            b.BrokerID,
            b.BrokerName,
            b.BrokerStatusCode,
            b.BrokerStatusDesc
        FROM
            TPCE_DM.dbo.D_Broker b WITH (NOLOCK)
    ) dm
FULL OUTER JOIN (
    SELECT
        hb.BrokerID,
        sb.BrokerName,
        sb.StatusCode,

```



```

        xst.StatusDesc
FROM
    dbo.HUB_Broker hb WITH (NOLOCK)
    JOIN dbo.SAT_Broker sb WITH (NOLOCK)
        ON hb.HUB_Broker_SQN = sb.HUB_Broker_SQN
    JOIN dbo.XREF_Status xst WITH (NOLOCK)
        ON xst.StatusCode = sb.StatusCode
WHERE
    sb.SAT_Broker_LEDTS IS NULL
) dv
    ON dm.BrokerID = dv.BrokerID
    AND dm.BrokerName = dv.BrokerName
    AND dm.BrokerStatusCode = dv.StatusCode
    AND dm.BrokerStatusDesc = dv.StatusDesc
WHERE
    dm.BrokerID IS NULL
    OR dv.BrokerID IS NULL;

```

```

/*****
CashType
*****/

```

```

USE TPCE_DV
GO

```

```

SELECT *
FROM
    (
        SELECT
            dct.IsCashFlag
        FROM
            TPCE_DM.dbo.D_CashType dct WITH (NOLOCK)
    ) dm
FULL OUTER JOIN
    (
        SELECT DISTINCT
            lt.IsCashFlag
        FROM
            TLNK_Trade lt WITH (NOLOCK)
    ) dv

```

```

        ON dm.IsCashFlag = dv.IsCashFlag
WHERE
    dm.IsCashFlag IS NULL
    OR dv.IsCashFlag IS NULL;

/*****
Company
*****/
USE TPCE_DV
GO

SELECT *
FROM
    (
        SELECT
            dco.CompanyID,
            CompanyName,
            CompanyDesc,
            CompanyStatusCode,
            CompanyStatusDesc,
            CompanyCEO,
            CompanyCreditRating,
            CompanyOpenDate,
            IndustryCode,
            IndustryDesc,
            SectorCode,
            SectorDesc,
            AddressLine1,
            AddressLine2,
            TownName,
            StateProvinceCode,
            PostalCode,
            CountryName
        FROM
            TPCE_DM.dbo.D_Company dco WITH (NOLOCK)
        WHERE
            dco.WH_CurrInd = 1
    ) dm
FULL OUTER JOIN

```

(

SELECT

```
sco.CompanyID,  
hco.CompanyName,  
sco.CompanyDesc,  
xst.StatusCode,  
xst.StatusDesc,  
scoceo.CompanyCEO,  
scocr.CompanyCreditRating,  
sco.CompanyOpenDate,  
xin.IndustryCode,  
xin.IndustryDesc,  
xsc.SectorCode,  
xsc.SectorDesc,  
sco.AddressLine1,  
ISNULL(sco.AddressLine2,'') AS AddressLine2, --This rule is implemented on the DM  
xt.TownName,  
xt.StateProvinceCode,  
xt.PostalCode,  
xt.CountryName
```

FROM

```
dbo.HUB_Company hco WITH (NOLOCK)  
JOIN dbo.SAT_Company sco WITH (NOLOCK)  
    ON sco.HUB_Company_SQN = hco.HUB_Company_SQN  
JOIN dbo.SAT_CompanyCEO scoceo WITH (NOLOCK)  
    ON scoceo.HUB_Company_SQN = hco.HUB_Company_SQN  
JOIN dbo.SAT_CompanyCreditRating scocr WITH (NOLOCK)  
    ON scocr.HUB_Company_SQN = hco.HUB_Company_SQN  
JOIN dbo.XREF_Status xst WITH (NOLOCK)  
    ON xst.StatusCode = sco.StatusCode  
JOIN dbo.XREF_Industry xin WITH (NOLOCK)  
    ON xin.IndustryCode = sco.IndustryCode  
JOIN dbo.XREF_Sector xsc WITH (NOLOCK)  
    ON xsc.SectorCode = sco.SectorCode  
JOIN dbo.XREF_Town xt WITH (NOLOCK)  
    ON xt.PostalCode = sco.PostalCode
```

WHERE

```
sco.SAT_Company_LEDTS IS NULL  
AND scoceo.SAT_CompanyCEO_LEDTS IS NULL  
AND scocr.SAT_CompanyCreditRating_LEDTS IS NULL
```

```

) dv
ON dm.CompanyID = dv.CompanyID
AND dm.CompanyName = dv.CompanyName
AND dm.CompanyDesc = dv.CompanyDesc
AND dm.CompanyStatusCode = dv.StatusCode
AND dm.CompanyStatusDesc = dv.StatusDesc
AND dm.CompanyCEO = dv.CompanyCEO
AND dm.CompanyCreditRating = dv.CompanyCreditRating
AND dm.CompanyOpenDate = dv.CompanyOpenDate
AND dm.IndustryCode = dv.IndustryCode
AND dm.IndustryDesc = dv.IndustryDesc
AND dm.SectorCode = dv.SectorCode
AND dm.SectorDesc = dv.SectorDesc
AND dm.AddressLine1 = dv.AddressLine1
AND dm.AddressLine2 = dv.AddressLine2
AND dm.TownName = dv.TownName
AND dm.StateProvinceCode = dv.StateProvinceCode
AND dm.PostalCode = dv.PostalCode
AND dm.CountryName = dv.CountryName

```

WHERE

```

dm.CompanyID IS NULL
OR dv.CompanyID IS NULL;

```

```

/*****
CustomerAccount
*****/

```

```

USE TPCE_DV;
GO

```

```

SELECT *
FROM

```

```

(
    SELECT
        AccountID,
        AccountName,
        TaxStatusCode,
        CustomerID,
        LastName,
        FirstName,

```

```

        MiddleInitial,
        GENDER,
        DateOfBirth,
        TierNumber,
        PhonePrimary,
        PhoneSecondary,
        EmailPrimary,
        EmailSecondary,
        AddressLine1,
        AddressLine2,
        TownName,
        StateProvinceCode,
        PostalCode,
        CountryName,
        CustomerStatusCode,
        CustomerStatusDesc
FROM
    TPCE_DM.dbo.D_CustomerAccount dca WITH (NOLOCK)
WHERE
    dca.WH_CurrInd = 1
) dm
FULL OUTER JOIN
(
    SELECT
        ha.AccountID,
        sa.AccountName,
        sa.TaxStatusCode,
        hc.CustomerID,
        sc.LastName,
        sc.FirstName,
        sc.MiddleInitial,
        CASE sc.GENDER
            WHEN 'M' THEN 'Male'
            WHEN 'F' THEN 'Female'
            ELSE 'Unknown'
        END as GENDER,
        sc.DateOfBirth,
        sc.TierNumber,
        CASE
            WHEN scc.PhoneCountryCodePrimary IS NULL

```

```

        AND scc.PhoneAreaCodePrimary IS NULL
        AND scc.PhoneLocalNumberPrimary IS NULL
        THEN 'Unknown'
    WHEN scc.PhoneExtensionPrimary IS NULL
        THEN scc.PhoneCountryCodePrimary + '-' + scc.PhoneAreaCodePrimary + '-' +
LEFT(scc.PhoneLocalNumberPrimary,3) + '-' + RIGHT(scc.PhoneLocalNumberPrimary,4)
    ELSE
        scc.PhoneCountryCodePrimary + '-' + scc.PhoneAreaCodePrimary + '-' +
LEFT(scc.PhoneLocalNumberPrimary,3) + '-' + RIGHT(scc.PhoneLocalNumberPrimary,4) + ' ext. ' + scc.PhoneExtensionPrimary
    END AS PhonePrimary,
    CASE
        WHEN scc.PhoneCountryCodeSecondary IS NULL
            AND scc.PhoneAreaCodeSecondary IS NULL
            AND scc.PhoneLocalNumberSecondary IS NULL
            THEN 'Unknown'
        WHEN scc.PhoneExtensionSecondary IS NULL
            THEN scc.PhoneCountryCodeSecondary + '-' + scc.PhoneAreaCodeSecondary + '-' +
LEFT(scc.PhoneLocalNumberSecondary,3) + '-' + RIGHT(scc.PhoneLocalNumberSecondary,4)
        ELSE
            scc.PhoneCountryCodeSecondary + '-' + scc.PhoneAreaCodeSecondary + '-' +
LEFT(scc.PhoneLocalNumberSecondary,3) + '-' + RIGHT(scc.PhoneLocalNumberSecondary,4) + ' ext. ' + scc.PhoneExtensionSecondary
    END AS PhoneSecondary,
    scc.EmailPrimary,
    scc.EmailSecondary,
    scc.AddressLine1,
    ISNULL(scc.AddressLine2,'') AS AddressLine2,
    xt.TownName,
    xt.StateProvinceCode,
    xt.PostalCode,
    xt.CountryName,
    xst.StatusCode,
    xst.StatusDesc
FROM
    dbo.HUB_Account ha WITH (NOLOCK)
    JOIN dbo.SAT_Account sa WITH (NOLOCK)
        ON sa.HUB_Account_SQN = ha.HUB_Account_SQN
    JOIN dbo.LNK_AccountCustomer lac WITH (NOLOCK)
        ON lac.HUB_Account_SQN = ha.HUB_Account_SQN
    JOIN dbo.HUB_Customer hc WITH (NOLOCK)
        ON hc.HUB_Customer_SQN = lac.HUB_Customer_SQN

```

```

JOIN dbo.SAT_Customer sc WITH (NOLOCK)
    ON sc.HUB_Customer_SQN = hc.HUB_Customer_SQN
JOIN dbo.SAT_CustomerContact scc WITH (NOLOCK)
    ON scc.HUB_Customer_SQN = hc.HUB_Customer_SQN
JOIN dbo.XREF_Status xst WITH (NOLOCK)
    ON xst.StatusCode = sc.StatusCode
JOIN dbo.XREF_Town xt WITH (NOLOCK)
    ON xt.PostalCode = scc.PostalCode
WHERE
    sa.SAT_Account_LEDTS IS NULL
    AND sc.SAT_Customer_LEDTS IS NULL
    AND scc.SAT_CustomerContact_LEDTS IS NULL
) dv
    ON dm.AccountID = dv.AccountID
    AND dm.AccountName = dv.AccountName
    AND dm.TaxStatusCode = dv.TaxStatusCode
    AND dm.CustomerID = dv.CustomerID
    AND dm.LastName = dv.LastName
    AND dm.FirstName = dv.FirstName
    AND dm.MiddleInitial = dv.MiddleInitial
    AND dm.GENDER = dv.GENDER
    AND dm.DateOfBirth = dv.DateOfBirth
    AND dm.TierNumber = dv.TierNumber
    AND dm.PhonePrimary = dv.PhonePrimary
    AND dm.PhoneSecondary = dv.PhoneSecondary
    AND dm.EmailPrimary = dv.EmailPrimary
    AND dm.EmailSecondary = dv.EmailSecondary
    AND dm.AddressLine1 = dv.AddressLine1
    AND dm.AddressLine2 = dv.AddressLine2
    AND dm.TownName = dv.TownName
    AND dm.StateProvinceCode = dv.StateProvinceCode
    AND dm.PostalCode = dv.PostalCode
    AND dm.CountryName = dv.CountryName
    AND dm.CustomerStatusCode = dv.StatusCode
    AND dm.CustomerStatusDesc = dv.StatusDesc
WHERE
    dm.AccountID IS NULL
    OR dv.AccountID IS NULL;

```

```

/*****
Exchange
*****/
USE TPCE_DV;
GO

SELECT *
FROM
(
    SELECT
        ExchangeID,
        ExchangeName,
        ExchangeDesc,
        OpeningTime,
        ClosingTime,
        ListedSecurities,
        AddressLine1,
        AddressLine2,
        TownName,
        StateProvinceCode,
        PostalCode,
        CountryName
    FROM
        TPCE_DM.dbo.D_Exchange WITH (NOLOCK)
) dm
FULL OUTER JOIN
(
    SELECT
        hex.ExchangeID,
        sex.ExchangeName,
        sex.ExchangeDesc,
        CASE LEN(sex.OpeningTime)
            WHEN 3 THEN '0' + LEFT(CONVERT(CHAR(3),sex.OpeningTime),1) + ':' +
RIGHT(CONVERT(CHAR(3),sex.OpeningTime),2)
            WHEN 4 THEN LEFT(CONVERT(CHAR(4),sex.OpeningTime),2) + ':' + RIGHT(CONVERT(CHAR(4),sex.OpeningTime),2)
        END AS OpeningTime,
        CASE LEN(sex.ClosingTime)
            WHEN 3 THEN '0' + LEFT(CONVERT(CHAR(3),sex.ClosingTime),1) + ':' +
RIGHT(CONVERT(CHAR(3),sex.ClosingTime),2)
            WHEN 4 THEN LEFT(CONVERT(CHAR(4),sex.ClosingTime),2) + ':' + RIGHT(CONVERT(CHAR(4),sex.ClosingTime),2)

```



```

        END AS ClosingTime,
        seks.ListedSecurities,
        sex.AddressLine1,
        ISNULL(sex.AddressLine2, '') AS AddressLine2,
        xt.TownName,
        xt.StateProvinceCode,
        xt.PostalCode,
        xt.CountryName
FROM
    dbo.HUB_Exchange hex WITH (NOLOCK)
JOIN    dbo.SAT_Exchange sex WITH (NOLOCK)
        ON sex.HUB_Exchange_SQN = hex.HUB_Exchange_SQN
JOIN    dbo.SAT_ExchangeSecurities seks WITH (NOLOCK)
        ON seks.HUB_Exchange_SQN = hex.HUB_Exchange_SQN
JOIN    dbo.XREF_Town xt WITH (NOLOCK)
        ON xt.PostalCode = sex.PostalCode
WHERE
    sex.SAT_Exchange_LEDTS IS NULL
    AND seks.SAT_ExchangeSecurities_LEDTS IS NULL
) dv
    ON dm.ExchangeID = dv.ExchangeID
    AND dm.ExchangeName = dv.ExchangeName
    AND dm.ExchangeDesc = dv.ExchangeDesc
    AND dm.OpeningTime = dv.OpeningTime
    AND dm.ClosingTime = dv.ClosingTime
    AND dm.ListedSecurities = dv.ListedSecurities
    AND dm.AddressLine1 = dv.AddressLine1
    AND dm.AddressLine2 = dv.AddressLine2
    AND dm.TownName = dv.TownName
    AND dm.StateProvinceCode = dv.StateProvinceCode
    AND dm.PostalCode = dv.PostalCode
    AND dm.CountryName = dv.CountryName
WHERE
    dm.ExchangeID IS NULL
    OR dv.ExchangeID IS NULL;

```

```

/*****
Security
*****/

```

```

USE TPCE_DV;
GO

SELECT *
FROM
(
    SELECT
        SecurityCode,
        SecurityName,
        IssueType,
        FirstTradeDate,
        SecurityStatusCode,
        SecurityStatusDesc
    FROM
        TPCE_DM.dbo.D_Security        WITH (NOLOCK)
) dm
FULL OUTER JOIN
(
    SELECT
        hs.SecurityCode,
        ss.SecurityName,
        ss.IssueType,
        ss.FirstTradeDate,
        xst.StatusCode,
        xst.StatusDesc
    FROM
        dbo.HUB_Security hs WITH (NOLOCK)
        JOIN dbo.SAT_Security ss WITH (NOLOCK)
            ON ss.HUB_Security_SQN = hs.HUB_Security_SQN
        JOIN dbo.XREF_Status xst WITH (NOLOCK)
            ON xst.StatusCode = ss.StatusCode
    WHERE
        ss.SAT_Security_LEDTS IS NULL
) dv
ON dm.SecurityCode = dv.SecurityCode
AND dm.SecurityName = dv.SecurityName
AND dm.IssueType = dv.IssueType
AND dm.FirstTradeDate = dv.FirstTradeDate
AND dm.SecurityStatusCode = dv.StatusCode
AND dm.SecurityStatusDesc = dv.StatusDesc

```

WHERE

dm.SecurityCode IS NULL  
OR dv.SecurityCode IS NULL;

/\*  
TradeStatus  
\*/

USE TPCE\_DV;  
GO

SELECT \*  
FROM

(  
    SELECT  
        dts.TradeStatusCode,  
        dts.TradeStatusDesc  
    FROM  
        TPCE\_DM.dbo.D\_TradeStatus dts WITH (NOLOCK)  
)  
dm  
FULL OUTER JOIN  
(  
    SELECT  
        xst.StatusCode,  
        xst.StatusDesc  
    FROM  
        dbo.XREF\_Status xst WITH (NOLOCK)  
)  
dv  
ON dm.TradeStatusCode = dv.StatusCode  
AND dm.TradeStatusDesc = dv.StatusDesc

WHERE

dm.TradeStatusCode IS NULL  
OR dv.StatusCode IS NULL;

/\*  
TradeType  
\*/

USE TPCE\_DV;  
GO

```

SELECT *
FROM
(
    SELECT
        dtt.TradeTypeCode,
        dtt.TradeTypeDesc,
        dtt.TradeCategory,
        dtt.MarketTransactionFlag
    FROM
        TPCE_DM.dbo.D_TradeType dtt WITH (NOLOCK)
) dm
FULL OUTER JOIN
(
    SELECT
        stt.TradeTypeCode,
        htt.TradeTypeDesc,
        CASE stt.IsSellFlag
            WHEN 0 THEN 'Buy'
            ELSE 'Sell'
        END AS TradeCategory,
        stt.IsMarketFlag
    FROM
        dbo.HUB_TradeType htt WITH (NOLOCK)
        JOIN dbo.SAT_TradeType stt WITH (NOLOCK)
            ON stt.HUB_TradeType_SQN = htt.HUB_TradeType_SQN
    WHERE
        stt.SAT_TradeType_LEDTS IS NULL
) dv
ON dm.TradeTypeCode = dv.TradeTypeCode
AND dm.TradeTypeDesc = dv.TradeTypeDesc
AND dm.TradeCategory = dv.TradeCategory
AND dm.MarketTransactionFlag = dv.IsMarketFlag
WHERE
    dm.TradeTypeCode IS NULL
    OR dv.TradeTypeCode IS NULL;

/*****
DailyMarketPerformance

```

\*\*\*\*\*/

```
USE TPCE_DV;  
GO
```

```
SELECT *  
FROM
```

```
(  
    SELECT  
        dd.[Date],  
        dco.CompanyName,  
        dex.ExchangeID,  
        ds.SecurityCode,  
        fdm.DayClose,  
        fdm.DayHigh,  
        fdm.DayLow,  
        fdm.DayVolume  
    FROM  
        TPCE_DM.dbo.F_DailyMarketPerformance fdm WITH (NOLOCK)  
    JOIN TPCE_DM.dbo.D_Date dd WITH (NOLOCK)  
        ON dd.Date_SK = fdm.Date_SK  
    JOIN TPCE_DM.dbo.D_Company dco WITH (NOLOCK)  
        ON dco.Company_SK = fdm.Company_SK  
    JOIN TPCE_DM.dbo.D_Exchange dex WITH (NOLOCK)  
        ON dex.Exchange_SK = fdm.Exchange_SK  
    JOIN TPCE_DM.dbo.D_Security ds WITH (NOLOCK)  
        ON ds.Security_SK = fdm.Security_SK  
)
```

```
) dm
```

```
FULL OUTER JOIN
```

```
(  
    SELECT  
        ldm.[Date],  
        hco.CompanyName,  
        hex.ExchangeID,  
        hs.SecurityCode,  
        ldm.DayClose,  
        ldm.DayHigh,  
        ldm.DayLow,  
        ldm.DayVolume  
    FROM  
        dbo.TLNK_DailyMarketPerformance ldm WITH (NOLOCK)
```

```

        JOIN dbo.HUB_Security hs WITH (NOLOCK)
            ON hs.HUB_Security_SQN = ldm.HUB_Security_SQN
        JOIN dbo.HUB_Company hco WITH (NOLOCK)
            ON hco.HUB_Company_SQN = ldm.HUB_Company_SQN
        JOIN dbo.HUB_Exchange hex WITH (NOLOCK)
            ON hex.HUB_Exchange_SQN = ldm.HUB_Exchange_SQN
    ) dv
    ON dm.[Date] = dv.[Date]
    AND dm.CompanyName = dv.CompanyName
    AND dm.ExchangeID = dv.ExchangeID
    AND dm.SecurityCode = dv.SecurityCode
    AND dm.DayClose = dv.DayClose
    AND dm.DayHigh = dv.DayHigh
    AND dm.DayLow = dv.DayLow
    AND dm.DayVolume = dv.DayVolume
WHERE
    dm.CompanyName IS NULL
    OR dv.CompanyName IS NULL;

/*****
QuarterlyCompanyPerformance
*****/
USE TPCE_DV;
GO

SELECT *
FROM
    (
        SELECT
            dq.FiscalQuarter,
            dq.FiscalYear,
            dco.CompanyName,
            dd.[Date],
            fcop.Revenue,
            fcop.NetEarnings,
            fcop.EarningsPerShareBasic,
            fcop.EarningsPerShareDiluted,
            fcop.Margin,
            fcop.Inventory,

```

```

        fcop.Assets,
        fcop.Liabilities,
        fcop.SharesOutstandingBasicAvg,
        fcop.SharesOutstandingDilutedAvg
FROM
    TPCE_DM.dbo.F_QuarterlyCompanyPerformance fcop WITH (NOLOCK)
    JOIN TPCE_DM.dbo.D_FiscalQuarter dq WITH (NOLOCK)
        ON dq.FiscalQuarter_SK = fcop.FiscalQuarter_SK
    JOIN TPCE_DM.dbo.D_Company dco WITH (NOLOCK)
        ON dco.Company_SK = fcop.Company_SK
    JOIN TPCE_DM.dbo.D_Date dd WITH (NOLOCK)
        ON dd.Date_SK = fcop.FiscalQuarterStartDate_SK
) dm
FULL OUTER JOIN
(
    SELECT
        tcop.FiscalQuarter,
        tcop.FiscalYear,
        hco.CompanyName,
        tcop.FiscalQuarterStartDate,
        tcop.Revenue,
        tcop.NetEarnings,
        tcop.EarningsPerShareBasic,
        tcop.EarningsPerShareDiluted,
        tcop.Margin,
        tcop.Inventory,
        tcop.Assets,
        tcop.Liabilities,
        tcop.SharesOutstandingBasicAvg,
        tcop.SharesOutstandingDilutedAvg
FROM
    dbo.TLTK_QuarterlyCompanyPerformance tcop WITH (NOLOCK)
    JOIN dbo.HUB_Company hco WITH (NOLOCK)
        ON hco.HUB_Company_SQN = tcop.HUB_Company_SQN
) dv
ON dm.FiscalQuarter = dv.FiscalQuarter
AND dm.FiscalYear = dv.FiscalYear
AND dm.CompanyName = dv.CompanyName
AND dm.[Date] = dv.FiscalQuarterStartDate
AND dm.Revenue = dv.Revenue

```

```

AND dm.NetEarnings = dv.NetEarnings
AND dm.EarningsPerShareBasic = dv.EarningsPerShareBasic
AND dm.EarningsPerShareDiluted = dv.EarningsPerShareDiluted
AND dm.Margin = dv.Margin
AND dm.Inventory = dv.Inventory
AND dm.Assets = dv.Assets
AND dm.Liabilities = dv.Liabilities
AND dm.SharesOutstandingBasicAvg = dv.SharesOutstandingBasicAvg
AND dm.SharesOutstandingDilutedAvg = dv.SharesOutstandingDilutedAvg

```

WHERE

```

dv.FiscalQuarter IS NULL
OR dm.FiscalQuarter IS NULL;

```

```

/*****
Trade
*****/

```

```

USE TPCE_DV;
GO

```

```

SELECT *
FROM

```

```

(
    SELECT
        dd.[Date],
        db.BrokerID,
        dca.AccountID,
        dco.CompanyName,
        dex.ExchangeID,
        ds.SecurityCode,
        dtt.TradeTypeDesc,
        dct.IsCashFlag,
        dts.TradeStatusCode,
        ft.TradeID,
        ft.TradeDateTime,
        ft.Quantity,
        ft.BidPrice,
        ft.TradePrice,
        ft.Charge,
        ft.Commission,

```



```

        ft.Tax
FROM
    TPCE_DM.dbo.F_Trade ft (NOLOCK)
    JOIN TPCE_DM.dbo.D_Date dd (NOLOCK)
        ON dd.Date_SK = ft.Date_SK
    JOIN TPCE_DM.dbo.D_Broker db (NOLOCK)
        ON db.Broker_SK = ft.Broker_SK
    JOIN TPCE_DM.dbo.D_CustomerAccount dca (NOLOCK)
        ON dca.CustomerAccount_SK = ft.CustomerAccount_SK
    JOIN TPCE_DM.dbo.D_Company dco (NOLOCK)
        ON dco.Company_SK = ft.Company_SK
    JOIN TPCE_DM.dbo.D_Exchange dex (NOLOCK)
        ON dex.Exchange_SK = ft.Exchange_SK
    JOIN TPCE_DM.dbo.D_Security ds (NOLOCK)
        ON ds.Security_SK = ft.Security_SK
    JOIN TPCE_DM.dbo.D_TradeType dtt (NOLOCK)
        ON dtt.TradeType_SK = ft.TradeType_SK
    JOIN TPCE_DM.dbo.D_CashType dct (NOLOCK)
        ON dct.CashType_SK = ft.CashType_SK
    JOIN TPCE_DM.dbo.D_TradeStatus dts (NOLOCK)
        ON dts.TradeStatus_SK = ft.TradeStatus_SK
) dm
FULL OUTER JOIN
(
    SELECT
        CONVERT(DATE,lt.TradeDateTime) AS [Date],
        hb.BrokerID,
        hac.AccountID,
        hco.CompanyName,
        hex.ExchangeID,
        hs.SecurityCode,
        htt.TradeTypeDesc,
        lt.IsCashFlag,
        lt.StatusCode,
        lt.TradeID,
        lt.TradeDateTime,
        lt.Quantity,
        lt.BidPrice,
        lt.TradePrice,
        lt.Charge,

```

```

        lt.Commission,
        lt.Tax
FROM
    dbo.TLNK_Trade lt (NOLOCK)
    JOIN dbo.HUB_Broker hb (NOLOCK)
        ON hb.HUB_Broker_SQN = lt.HUB_Broker_SQN
    JOIN dbo.HUB_Account hac (NOLOCK)
        ON hac.HUB_Account_SQN = lt.HUB_Account_SQN
    JOIN dbo.HUB_Company hco (NOLOCK)
        ON hco.HUB_Company_SQN = lt.HUB_Company_SQN
    JOIN dbo.HUB_Exchange hex (NOLOCK)
        ON hex.HUB_Exchange_SQN = lt.HUB_Exchange_SQN
    JOIN dbo.HUB_Security hs (NOLOCK)
        ON hs.HUB_Security_SQN = lt.HUB_Security_SQN
    JOIN dbo.HUB_TradeType htt (NOLOCK)
        ON htt.HUB_TradeType_SQN = lt.HUB_TradeType_SQN
) dv
    ON dm.[Date] = dv.[Date]
    AND dm.BrokerID = dv.BrokerID
    AND dm.AccountID = dv.AccountID
    AND dm.CompanyName = dv.CompanyName
    AND dm.ExchangeID = dv.ExchangeID
    AND dm.SecurityCode = dv.SecurityCode
    AND dm.TradeTypeDesc = dv.TradeTypeDesc
    AND dm.IsCashFlag = dv.IsCashFlag
    AND dm.TradeStatusCode = dv.StatusCode
    AND dm.TradeID = dv.TradeID
    AND dm.TradeDateTime = dv.TradeDateTime
    AND dm.Quantity = dv.Quantity
    AND dm.BidPrice = dv.BidPrice
    AND dm.TradePrice = dv.TradePrice
    AND dm.Charge = dv.Charge
    AND dm.Commission = dv.Commission
    AND dm.Tax = dv.Tax
WHERE
    dm.CompanyName IS NULL
    OR dv.CompanyName IS NULL;

```

## 1.4 Query Performance Scripts

### 1.4.1 Dimensional Model

```

/*****
Category 1 - Company
*****/

USE TPCE_DM;
GO

SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT
    dco.CompanyID AS CompanyID,
    dco.CompanyName AS CompanyName,
    dco.CompanyDesc AS CompanyDesc,
    dco.CompanyStatusCode AS CompanyStatusCode,
    dco.CompanyStatusDesc AS CompanyStatusDesc,
    dco.CompanyCEO AS CompanyCEO,
    dco.CompanyCreditRating AS CompanyCreditRating,
    dco.CompanyOpenDate AS CompanyOpenDate,
    dco.IndustryCode AS IndustryCode,
    dco.IndustryDesc AS IndustryDesc,
    dco.SectorCode AS SectorCode,
    dco.SectorDesc AS SectorDesc,
    dco.AddressLine1 AS AddressLine1,
    dco.AddressLine2 AS AddressLine2,
    dco.TownName AS TownName,
```

```

        dco.StateProvinceCode AS StateProvinceCode,
        dco.PostalCode AS PostalCode,
        dco.CountryName AS CountryName
INTO #temp
FROM      TPCE_DM.dbo.D_Company dco
WHERE
        dco.WH_CurrInd =1

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('1 - Company',@Iteration,'DM',@QueryStart,@QueryEnd,@RecordCount);

```

```

/*****
Category 1 - CustomerAccount
*****/

```

```

USE TPCE_DM;
GO

```

```

SET NOCOUNT ON;

```

```

DECLARE
        @QueryStart AS DATETIME2(7),
        @QueryEnd AS DATETIME2(7),
        @Iteration AS INT,
        @RecordCount AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

```

```

SELECT
        dca.AccountID AS AccountID,
        dca.AccountName AS AccountName,
        dca.TaxStatusCode AS TaxStatusCode,
        dca.TaxStatusDesc AS TaxStatusDesc,

```

```

dca.CustomerID AS CustomerID,
dca.LastName AS LastName,
dca.FirstName AS FirstName,
dca.MiddleInitial AS MiddleInitial,
dca.Gender AS Gender,
dca.DateOfBirth AS DateOfBirth,
dca.TierNumber AS TierNumber,
dca.TierDesc AS TierDesc,
dca.PhonePrimary AS PhonePrimary,
dca.PhoneSecondary AS PhoneSecondary,
dca.EmailPrimary AS EmailPrimary,
dca.EmailSecondary AS EmailSecondary,
dca.AddressLine1 AS AddressLine1,
dca.AddressLine2 AS AddressLine2,
dca.TownName AS TownName,
dca.StateProvinceCode AS StateProvinceCode,
dca.PostalCode AS PostalCode,
dca.CountryName AS CountryName,
dca.CustomerStatusCode AS CustomerStatusCode,
dca.CustomerStatusDesc AS CustomerStatusDesc
INTO #temp
FROM TPCE_DM.dbo.D_CustomerAccount dca
WHERE dca.WH_CurrInd = 1;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('1 - CustomerAccount',@Iteration,'DM',@QueryStart,@QueryEnd,@RecordCount);

/*****
Category 1 - Security
*****/

USE TPCE_DM;
GO

```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7),  
    @Iteration AS INT,  
    @RecordCount AS INT;
```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
```

```
SET @QueryStart = SYSDATETIME();
```

```
SELECT
```

```
    SecurityCode,  
    SecurityName,  
    IssueType,  
    FirstTradeDate,  
    SecurityStatusCode,  
    SecurityStatusDesc
```

```
INTO #temp
```

```
FROM
```

```
    TPCE_DM.dbo.D_Security;
```

```
SET @RecordCount = @@ROWCOUNT;
```

```
SET @QueryEnd = SYSDATETIME();
```

```
INSERT INTO master.dbo.QueryResults
```

```
VALUES ('1 - Security',@Iteration,'DM',@QueryStart,@QueryEnd,@RecordCount);
```

```
/*  
Category 2 - Company  
*/
```

```
USE TPCE_DM;
```

```
GO
```

```
SET NOCOUNT ON;
```

```

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT
    dco.SectorCode,
    dco.SectorDesc,
    dco.IndustryCode,
    dco.IndustryDesc,
    COUNT(*) AS cnt
INTO #temp
FROM
    dbo.D_Company dco
WHERE
    dco.TownName = 'New York'
    AND dco.WH_CurrInd = 1
GROUP BY
    dco.SectorCode,
    dco.SectorDesc,
    dco.IndustryCode,
    dco.IndustryDesc
HAVING
    COUNT(*) >= 10
ORDER BY
    COUNT(*) DESC;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('2 - Company',@Iteration,'DM',@QueryStart,@QueryEnd,@RecordCount);

```

```

/*****

```

## Category 2 - CustomerAccount

\*\*\*\*\*/

```
USE TPCE_DM;  
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
```

```
    @QueryStart AS DATETIME2(7),  
    @QueryEnd AS DATETIME2(7),  
    @Iteration AS INT,  
    @RecordCount AS INT;
```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;  
SET @QueryStart = SYSDATETIME();
```

```
SELECT DISTINCT
```

```
    dca.LastName + ', ' + dca.FirstName AS FullName,  
    dca.TierNumber AS TierNumber,  
    dca.TierDesc AS TierDesc,  
    dca.PhonePrimary AS PhonePrimary,  
    dca.EmailPrimary AS EmailPrimary
```

```
INTO #temp
```

```
FROM
```

```
    TPCE_DM.dbo.D_CustomerAccount dca
```

```
WHERE
```

```
    dca.Gender = 'Male'  
    AND dca.CustomerStatusCode = 'ACTV'  
    AND dca.WH_CurrInd = 1
```

```
ORDER BY
```

```
    dca.TierNumber DESC;
```

```
SET @RecordCount = @@ROWCOUNT;
```

```
SET @QueryEnd = SYSDATETIME();
```

```
INSERT INTO master.dbo.QueryResults
```

```
VALUES ('2 - CustomerAccount', @Iteration, 'DM', @QueryStart, @QueryEnd, @RecordCount);
```



```

/*****
Category 2 - Security
*****/

USE TPCE_DM;
GO

SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT
    ds.IssueType AS IssueType,
    COUNT(*) AS NumberOfSecurities
INTO #tmp
FROM
    dbo.D_Security ds
WHERE
    ds.SecurityStatusCode = 'ACTV'
    AND ds.FirstTradeDate >= DATEADD(YEAR, -50, GETDATE())
GROUP BY
    ds.IssueType
ORDER BY
    ds.IssueType;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('2 - Security', @Iteration, 'DM', @QueryStart, @QueryEnd, @RecordCount);

```

```

/*****
Category 3 - CustomerBroker
*****/

```

```

USE TPCE_DM;
GO

```

```

SET NOCOUNT ON;

```

```

DECLARE

```

```

    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

```

```

SELECT DISTINCT

```

```

    db.BrokerName AS BrokerName,
    dca.LastName + ', ' + dca.FirstName AS FullName,
    dca.PhonePrimary AS PhonePrimary,
    dca.EmailPrimary AS EmailPrimary

```

```

INTO #temp
FROM

```

```

    dbo.D_CustomerAccount dca
    JOIN dbo.F_Trade ft
        ON ft.CustomerAccount_SK = dca.CustomerAccount_SK
    JOIN dbo.D_Broker db
        ON db.Broker_SK = ft.Broker_SK

```

```

WHERE

```

```

    db.BrokerStatusCode = 'ACTV'
    AND dca.CustomerStatusCode = 'ACTV'
    AND dca.WH_CurrInd = 1

```

```

ORDER BY

```

```

    db.BrokerName,
    dca.LastName + ', ' + dca.FirstName;

```

```

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('3 - CustomerBroken',@Iteration,'DM',@QueryStart,@QueryEnd,@RecordCount);

```

```

/*****
Category 4 - DailyMarketPerformance
*****/

```

```

USE TPCE_DM;
GO

```

```

SET NOCOUNT ON;

```

```

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

```

```

SELECT
    dd.[Date] AS [Date],
    dco.CompanyName AS CompanyName,
    dex.ExchangeID AS ExchangeID,
    ds.SecurityCode AS SecurityCode,
    fdm.DayClose AS DayClose,
    fdm.DayHigh AS DayHigh,
    fdm.DayLow AS DayLow,
    fdm.DayVolume AS DayVolume
INTO #temp
FROM
    TPCE_DM.dbo.F_DailyMarketPerformance fdm
    JOIN TPCE_DM.dbo.D_Date dd
        ON dd.Date_SK = fdm.Date_SK

```

```

JOIN TPCE_DM.dbo.D_Company dco
    ON dco.Company_SK = fdm.Company_SK
JOIN TPCE_DM.dbo.D_Exchange dex
    ON dex.Exchange_SK = fdm.Exchange_SK
JOIN TPCE_DM.dbo.D_Security ds
    ON ds.Security_SK = fdm.Security_SK
WHERE
    ds.SecurityCode IN ( 'A', 'AA', 'PCTI-a', 'PCAR-b', 'NW-d', 'CMC-c', 'CNNPRA-b', 'AZO-b', 'NI-e', 'GGTPR-b', 'TERN-c', 'BNGPRC-b', 'TFX-
d', 'EVOLPRB', 'TXNPRB-c', 'MCKPRC', 'CYLK-e', 'DQEPRB-b', 'BSET-c', 'CHBPRD-d'
)
ORDER BY
    ds.SecurityCode;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ( '4 - DailyMarketPerformance', @Iteration, 'DM', @QueryStart, @QueryEnd, @RecordCount);

/*****
Category 4 - Trade
*****/

USE TPCE_DM;
GO

SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT

```

```
dd.[Date],
db.BrokerID,
dca.AccountID,
dco.CompanyName,
dex.ExchangeID,
ds.SecurityCode,
dtt.TradeTypeDesc,
dct.IsCashFlag,
dts.TradeStatusCode,
ft.TradeID,
ft.TradeDateTime,
ft.Quantity,
ft.BidPrice,
ft.TradePrice,
ft.Charge,
ft.Commission,
ft.Tax
```

```
INTO #temp
```

```
FROM
```

```
TPCE_DM.dbo.F_Trade ft (NOLOCK)
JOIN TPCE_DM.dbo.D_Date dd (NOLOCK)
    ON dd.Date_SK = ft.Date_SK
JOIN TPCE_DM.dbo.D_Broker db (NOLOCK)
    ON db.Broker_SK = ft.Broker_SK
JOIN TPCE_DM.dbo.D_CustomerAccount dca (NOLOCK)
    ON dca.CustomerAccount_SK = ft.CustomerAccount_SK
JOIN TPCE_DM.dbo.D_Company dco (NOLOCK)
    ON dco.Company_SK = ft.Company_SK
JOIN TPCE_DM.dbo.D_Exchange dex (NOLOCK)
    ON dex.Exchange_SK = ft.Exchange_SK
JOIN TPCE_DM.dbo.D_Security ds (NOLOCK)
    ON ds.Security_SK = ft.Security_SK
JOIN TPCE_DM.dbo.D_TradeType dtt (NOLOCK)
    ON dtt.TradeType_SK = ft.TradeType_SK
JOIN TPCE_DM.dbo.D_CashType dct (NOLOCK)
    ON dct.CashType_SK = ft.CashType_SK
JOIN TPCE_DM.dbo.D_TradeStatus dts (NOLOCK)
    ON dts.TradeStatus_SK = ft.TradeStatus_SK
```

```
WHERE
```

```
dd.[Date] = '2005/01/03';
```

```
SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('4 - Trade',@Iteration,'DM',@QueryStart,@QueryEnd,@RecordCount);
```

```
/******
Category 5 - QuarterlyCompanyPerformance
******/
```

```
USE TPCE_DM;
GO
```

```
SET NOCOUNT ON;
```

```
DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;
```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();
```

```
SELECT
    dq.FiscalQuarter,
    dq.FiscalYear,
    dco.CompanyName,
    dd.[Date],
    fcop.Revenue,
    fcop.NetEarnings,
    fcop.EarningsPerShareBasic,
    fcop.EarningsPerShareDiluted,
    fcop.Margin,
    fcop.Inventory,
    fcop.Assets,
    fcop.Liabilities,
```

```

        fcop.SharesOutstandingBasicAvg,
        fcop.SharesOutstandingDilutedAvg
    INTO #temp
    FROM
        TPCE_DM.dbo.F_QuarterlyCompanyPerformance fcop
    JOIN TPCE_DM.dbo.D_FiscalQuarter dq
        ON dq.FiscalQuarter_SK = fcop.FiscalQuarter_SK
    JOIN TPCE_DM.dbo.D_Company dco
        ON dco.Company_SK = fcop.Company_SK
    JOIN TPCE_DM.dbo.D_Date dd
        ON dd.Date_SK = fcop.FiscalQuarterStartDate_SK
    WHERE
        dq.FiscalYear = 2004
        AND fcop.Assets >= (fcop.Liabilities * 1.5)
        AND fcop.EarningsPerShareBasic > 0.25;

    SET @RecordCount = @@ROWCOUNT;
    SET @QueryEnd = SYSDATETIME();

    INSERT INTO master.dbo.QueryResults
    VALUES ( '5 - QuarterlyCompanyPerformance', @Iteration, 'DM', @QueryStart, @QueryEnd, @RecordCount);

/*****
Category 5 - Trade
*****/

USE TPCE_DM;
GO

SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

    SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;

```

```

SET @QueryStart = SYSDATETIME();

SELECT
    dd.[Year] AS [Year],
    dd.[MonthName] + ' ' + CONVERT(VARCHAR(4),dd.[Year]) AS [Month],
    db.BrokerName AS BrokerName,
    SUM(ft.Quantity) AS Quantity,
    SUM(ft.TradePrice * ft.Quantity) AS SalesAmount,
    SUM(ft.Commission) AS Commission
INTO #temp
FROM
    TPCE_DM.dbo.F_Trade ft (NOLOCK)
    JOIN TPCE_DM.dbo.D_Date dd (NOLOCK)
        ON dd.Date_SK = ft.Date_SK
    JOIN TPCE_DM.dbo.D_Broker db (NOLOCK)
        ON db.Broker_SK = ft.Broker_SK
WHERE
    dd.[Year] = 2005
GROUP BY
    dd.[Year],
    dd.[MonthName] + ' ' + CONVERT(VARCHAR(4),dd.[Year]),
    db.BrokerName;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ( '5 - Trade',@Iteration,'DM',@QueryStart,@QueryEnd,@RecordCount);

```

## 1.4.2 Data Vault Model

```

/*****
Category 1 - Company
*****/

SET NOCOUNT ON;

```



DECLARE

@QueryStart AS DATETIME2(7),  
@QueryEnd AS DATETIME2(7),  
@Iteration AS INT,  
@RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;

SET @QueryStart = SYSDATETIME();

SELECT

sco.CompanyID AS CompanyID,  
hco.CompanyName AS CompanyName,  
sco.CompanyDesc AS CompanyDesc,  
xst.StatusCode AS StatusCode,  
xst.StatusDesc AS StatusDesc,  
scoceo.CompanyCEO AS CompanyCEO,  
scocr.CompanyCreditRating AS CompanyCreditRating,  
sco.CompanyOpenDate AS CompanyOpenDate,  
xin.IndustryCode AS IndustryCode,  
xin.IndustryDesc AS IndustryDesc,  
xsc.SectorCode AS SectorCode,  
xsc.SectorDesc AS SectorDesc,  
sco.AddressLine1 AS AddressLine1,  
ISNULL(sco.AddressLine2, '') AS AddressLine2,  
xt.TownName AS TownName,  
xt.StateProvinceCode AS StateProvinceCode,  
xt.PostalCode AS PostalCode,  
xt.CountryName AS CountryName

INTO #temp

FROM

dbo.HUB\_Company hco  
JOIN dbo.SAT\_Company sco  
ON sco.HUB\_Company\_SQN = hco.HUB\_Company\_SQN  
JOIN dbo.SAT\_CompanyCEO scoceo  
ON scoceo.HUB\_Company\_SQN = hco.HUB\_Company\_SQN  
JOIN dbo.SAT\_CompanyCreditRating scocr  
ON scocr.HUB\_Company\_SQN = hco.HUB\_Company\_SQN  
JOIN dbo.XREF\_Status xst  
ON xst.StatusCode = sco.StatusCode

```

JOIN dbo.XREF_Industry xin
    ON xin.IndustryCode = sco.IndustryCode
JOIN dbo.XREF_Sector xsc
    ON xsc.SectorCode = sco.SectorCode
JOIN dbo.XREF_Town xt
    ON xt.PostalCode = sco.PostalCode
WHERE
    sco.SAT_Company_LEDTS IS NULL
    AND scoceo.SAT_CompanyCEO_LEDTS IS NULL
    AND scocr.SAT_CompanyCreditRating_LEDTS IS NULL;

```

```

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

```

```

INSERT INTO master.dbo.QueryResults
VALUES ('1 - Company',@Iteration,'DV',@QueryStart,@QueryEnd,@RecordCount);

```

```

/*****
Category 1 - CustomerAccount
*****/

```

```

SET NOCOUNT ON;

```

```

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

```

```

SELECT
    ha.AccountID AS AccountID,
    sa.AccountName AS AccountName,
    sa.TaxStatusCode AS TaxStatusCode,
    CASE sa.TaxStatusCode
        WHEN 0 THEN 'Not taxable.'

```

```

        WHEN 1 THEN 'Taxable. Withhold tax.'
        WHEN 2 THEN 'Taxable. Do not withhold tax.'
        ELSE 'Unknown'
    END AS TaxStatusDesc,
    hc.CustomerID AS CustomerID,
    sc.LastName AS LastName,
    sc.FirstName AS FirstName,
    sc.MiddleInitial AS MiddleInitial,
    CASE sc.Gender
        WHEN 'M' THEN 'Male'
        WHEN 'F' THEN 'Female'
        ELSE 'Unknown'
    END AS Gender,
    sc.DateOfBirth AS DateOfBirth,
    sc.TierNumber AS TierNumber,
    CASE sc.TierNumber
        WHEN 1 THEN 'Has between 1 and 4 accounts.'
        WHEN 2 THEN 'Has between 2 and 8 accounts.'
        WHEN 3 THEN 'Has between 5 and 10 accounts.'
        ELSE 'Unknown'
    END AS TierDesc,
    CASE
        WHEN scc.PhoneCountryCodePrimary IS NULL
            AND scc.PhoneAreaCodePrimary IS NULL
            AND scc.PhoneLocalNumberPrimary IS NULL
        THEN 'Unknown'
        WHEN scc.PhoneExtensionPrimary IS NULL
        THEN scc.PhoneCountryCodePrimary + '-' + scc.PhoneAreaCodePrimary + '-' + LEFT(scc.PhoneLocalNumberPrimary,3) +
        '-' + RIGHT(scc.PhoneLocalNumberPrimary,4)
        ELSE
            scc.PhoneCountryCodePrimary + '-' + scc.PhoneAreaCodePrimary + '-' + LEFT(scc.PhoneLocalNumberPrimary,3) + '-' +
        RIGHT(scc.PhoneLocalNumberPrimary,4) + ' ext. ' + scc.PhoneExtensionPrimary
    END AS PhonePrimary,
    CASE
        WHEN scc.PhoneCountryCodeSecondary IS NULL
            AND scc.PhoneAreaCodeSecondary IS NULL
            AND scc.PhoneLocalNumberSecondary IS NULL
        THEN 'Unknown'
        WHEN scc.PhoneExtensionSecondary IS NULL

```

```

        THEN scc.PhoneCountryCodeSecondary + '-' + scc.PhoneAreaCodeSecondary + '-' +
LEFT(scc.PhoneLocalNumberSecondary,3) + '-' + RIGHT(scc.PhoneLocalNumberSecondary,4)
        ELSE
        scc.PhoneCountryCodeSecondary + '-' + scc.PhoneAreaCodeSecondary + '-' + LEFT(scc.PhoneLocalNumberSecondary,3) +
        '-' + RIGHT(scc.PhoneLocalNumberSecondary,4) + ' ext. ' + scc.PhoneExtensionSecondary
    END AS PhoneSecondary,
    scc.EmailPrimary AS EmailPrimary,
    scc.EmailSecondary AS EmailSecondary,
    scc.AddressLine1 AS AddressLine1,
    ISNULL(scc.AddressLine2, '') AS AddressLine2,
    xt.TownName AS TownName,
    xt.StateProvinceCode AS StateProvinceCode,
    xt.PostalCode AS PostalCode,
    xt.CountryName AS CountryName,
    xst.StatusCode AS CustomerStatusCode,
    xst.StatusDesc AS CustomerStatusDesc
INTO #temp
FROM
    dbo.HUB_Account ha
    JOIN dbo.SAT_Account sa
        ON sa.HUB_Account_SQN = ha.HUB_Account_SQN
    JOIN dbo.LNK_AccountCustomer lac
        ON lac.HUB_Account_SQN = ha.HUB_Account_SQN
    JOIN dbo.SAT_AccountCustomerStatus sacs
        ON sacs.LNK_AccountCustomer_SQN = lac.LNK_AccountCustomer_SQN
    JOIN dbo.HUB_Customer hc
        ON hc.HUB_Customer_SQN = lac.HUB_Customer_SQN
    JOIN dbo.SAT_Customer sc
        ON sc.HUB_Customer_SQN = hc.HUB_Customer_SQN
    JOIN dbo.SAT_CustomerContact scc
        ON scc.HUB_Customer_SQN = hc.HUB_Customer_SQN
    JOIN dbo.XREF_Status xst
        ON xst.StatusCode = sc.StatusCode
    JOIN dbo.XREF_Town xt
        ON xt.PostalCode = sc.PostalCode
WHERE
    sacs.SAT_AccountCustomerStatus_LEDTS IS NULL
    AND sa.SAT_Account_LEDTS IS NULL
    AND sc.SAT_Customer_LEDTS IS NULL
    AND scc.SAT_CustomerContact_LEDTS IS NULL;

```

```

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('1 - CustomerAccount',@Iteration,'DV',@QueryStart,@QueryEnd,@RecordCount);

/*****
Category 1 - Security
*****/

SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT
    hs.SecurityCode,
    ss.SecurityName,
    ss.IssueType,
    ss.FirstTradeDate,
    xst.StatusCode,
    xst.StatusDesc
INTO #temp
FROM
    dbo.HUB_Security hs
    JOIN dbo.SAT_Security ss
        ON ss.HUB_Security_SQN = hs.HUB_Security_SQN
    JOIN dbo.XREF_Status xst
        ON xst.StatusCode = ss.StatusCode
WHERE
    ss.SAT_Security_LEDTS IS NULL;

```

```

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('1 - Security',@Iteration,'DV',@QueryStart,@QueryEnd,@RecordCount);

```

```

/*****
Category 2 - Company
*****/

```

```

SET NOCOUNT ON;

```

```

DECLARE

```

```

    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

```

```

SELECT

```

```

    xsc.SectorCode AS SectorCode,
    xsc.SectorDesc AS SectorDesc,
    xin.IndustryCode AS IndustryCode,
    xin.IndustryDesc AS IndustryDesc,
    COUNT(*) AS cnt

```

```

INTO #temp

```

```

FROM

```

```

    dbo.HUB_Company hco
JOIN  dbo.SAT_Company sco
      ON sco.HUB_Company_SQN = hco.HUB_Company_SQN
JOIN  dbo.XREF_Industry xin
      ON xin.IndustryCode = sco.IndustryCode
JOIN  dbo.XREF_Sector xsc
      ON xsc.SectorCode = sco.SectorCode

```

```

        JOIN dbo.XREF_Town xt
            ON xt.PostalCode = sco.PostalCode
WHERE
    sco.SAT_Company_LEDTS IS NULL
    AND xt.TownName = 'New York'
GROUP BY
    xsc.SectorCode,
    xsc.SectorDesc,
    xin.IndustryCode,
    xin.IndustryDesc
HAVING
    COUNT(*) >= 10
ORDER BY
    COUNT(*) DESC;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('2 - Company',@Iteration,'DV',@QueryStart,@QueryEnd,@RecordCount);

/*****
Category 2 - CustomerAccount
*****/
SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT
    sc.LastName + ', ' + sc.FirstName AS FullName,
    sc.TierNumber AS TierNumber,
    CASE sc.TierNumber

```

```

        WHEN 1 THEN 'Has between 1 and 4 accounts.'
        WHEN 2 THEN 'Has between 2 and 8 accounts.'
        WHEN 3 THEN 'Has between 5 and 10 accounts.'
        ELSE 'Unknown'
    END AS TierDesc,
    CASE
        WHEN scc.PhoneCountryCodePrimary IS NULL
            AND scc.PhoneAreaCodePrimary IS NULL
            AND scc.PhoneLocalNumberPrimary IS NULL
            THEN 'Unknown'
        WHEN scc.PhoneExtensionPrimary IS NULL
            THEN scc.PhoneCountryCodePrimary + '-' + scc.PhoneAreaCodePrimary + '-' + LEFT(scc.PhoneLocalNumberPrimary,3) +
            '-' + RIGHT(scc.PhoneLocalNumberPrimary,4)
        ELSE
            scc.PhoneCountryCodePrimary + '-' + scc.PhoneAreaCodePrimary + '-' + LEFT(scc.PhoneLocalNumberPrimary,3) + '-' +
            RIGHT(scc.PhoneLocalNumberPrimary,4) + ' ext. ' + scc.PhoneExtensionPrimary
        END AS PhonePrimary,
        scc.EmailPrimary AS EmailPrimary
    INTO #temp
    FROM
        dbo.HUB_Customer hc
        JOIN dbo.SAT_Customer sc
            ON sc.HUB_Customer_SQN = hc.HUB_Customer_SQN
        JOIN dbo.SAT_CustomerContact scc
            ON scc.HUB_Customer_SQN = hc.HUB_Customer_SQN
    WHERE
        sc.Gender = 'M'
        AND sc.StatusCode = 'ACTV'
        AND sc.SAT_Customer_LEDTS IS NULL
        AND scc.SAT_CustomerContact_LEDTS IS NULL
    ORDER BY
        sc.TierNumber DESC;

    SET @RecordCount = @@ROWCOUNT;
    SET @QueryEnd = SYSDATETIME();

    INSERT INTO master.dbo.QueryResults
    VALUES ('2 - CustomerAccount',@Iteration,'DV',@QueryStart,@QueryEnd,@RecordCount);

```



```

/*****
Category 2 - Security
*****/

SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT
    ss.IssueType AS IssueType,
    COUNT(*) AS NumberOfSecurities
INTO #temp
FROM
    dbo.HUB_Security hs
    JOIN dbo.SAT_Security ss
        ON ss.HUB_Security_SQN = hs.HUB_Security_SQN
WHERE
    ss.SAT_Security_LEDTS IS NULL
    AND ss.StatusCode = 'ACTV'
    AND ss.FirstTradeDate >= DATEADD(YEAR, -50, GETDATE())
GROUP BY
    ss.IssueType
ORDER BY
    ss.IssueType;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('2 - Security', @Iteration, 'DV', @QueryStart, @QueryEnd, @RecordCount);

```

```

/*****
Category 3 - CustomerBroker
*****/

SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT DISTINCT
    sb.BrokerName AS BrokerName,
    sc.LastName + ', ' + sc.FirstName AS FullName,
    CASE
        WHEN scc.PhoneCountryCodePrimary IS NULL
            AND scc.PhoneAreaCodePrimary IS NULL
            AND scc.PhoneLocalNumberPrimary IS NULL
            THEN 'Unknown'
        WHEN scc.PhoneExtensionPrimary IS NULL
            THEN scc.PhoneCountryCodePrimary + '-' + scc.PhoneAreaCodePrimary + '-' + LEFT(scc.PhoneLocalNumberPrimary,3) +
            '-' + RIGHT(scc.PhoneLocalNumberPrimary,4)
        ELSE
            scc.PhoneCountryCodePrimary + '-' + scc.PhoneAreaCodePrimary + '-' + LEFT(scc.PhoneLocalNumberPrimary,3) + '-' +
            RIGHT(scc.PhoneLocalNumberPrimary,4) + ' ext. ' + scc.PhoneExtensionPrimary
        END AS PhonePrimary,
    scc.EmailPrimary
INTO #temp
FROM
    dbo.HUB_Customer hc
JOIN dbo.SAT_Customer sc
    ON sc.HUB_Customer_SQN = hc.HUB_Customer_SQN
JOIN dbo.SAT_CustomerContact scc
    ON scc.HUB_Customer_SQN = hc.HUB_Customer_SQN

```

```

JOIN dbo.LNK_AccountCustomer lac
    ON lac.HUB_Customer_SQN = hc.HUB_Customer_SQN
JOIN dbo.HUB_Account ha
    ON ha.HUB_Account_SQN = lac.HUB_Account_SQN
JOIN dbo.LNK_AccountBroker lab
    ON lab.HUB_Account_SQN = ha.HUB_Account_SQN
JOIN dbo.HUB_Broker hb
    ON hb.HUB_Broker_SQN = lab.HUB_Broker_SQN
JOIN dbo.SAT_Broker sb
    ON sb.HUB_Broker_SQN = hb.HUB_Broker_SQN
WHERE
    sb.StatusCode = 'ACTV'
    AND sc.StatusCode = 'ACTV'
    AND sb.SAT_Broker_LEDTS IS NULL
    AND sc.SAT_Customer_LEDTS IS NULL;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('3 - CustomerBroker',@Iteration,'DV',@QueryStart,@QueryEnd,@RecordCount);

/*****
Category 4 - DailyMarketPerformance
*****/

SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT
    ldm.[Date] AS [Date],

```

```

        hco.CompanyName AS CompanyName,
        hex.ExchangeID AS ExchangeID,
        hs.SecurityCode AS SecurityCode,
        ldm.DayClose AS DayClose,
        ldm.DayHigh AS DayHigh,
        ldm.DayLow AS DayLow,
        ldm.DayVolume AS DayVolume
    INTO #temp
    FROM
        dbo.TLNK_DailyMarketPerformance ldm
    JOIN  dbo.HUB_Security hs
        ON hs.HUB_Security_SQN = ldm.HUB_Security_SQN
    JOIN  dbo.HUB_Company hco
        ON hco.HUB_Company_SQN = ldm.HUB_Company_SQN
    JOIN  dbo.HUB_Exchange hex
        ON hex.HUB_Exchange_SQN = ldm.HUB_Exchange_SQN
    WHERE
        hs.SecurityCode IN ('A', 'AA', 'PCTI-a', 'PCAR-b', 'NW-d', 'CMC-c', 'CNNPRA-b', 'AZO-b', 'NI-e', 'GGTPR-b', 'TERN-c', 'BNGPRC-b', 'TFX-
        d', 'EVOLPRB', 'TXNPRB-c', 'MCKPRC', 'CYLK-e', 'DQEPRB-b', 'BSET-c', 'CHBPRD-d')
    ORDER BY
        hs.SecurityCode;

    SET @RecordCount = @@ROWCOUNT;
    SET @QueryEnd = SYSDATETIME();

    INSERT INTO master.dbo.QueryResults
    VALUES ('4 - DailyMarketPerformance', @Iteration, 'DV', @QueryStart, @QueryEnd, @RecordCount);

    /*****
    Category 4 - Trade
    *****/

    SET NOCOUNT ON;

    DECLARE
        @QueryStart AS DATETIME2(7),
        @QueryEnd AS DATETIME2(7),
        @Iteration AS INT,
        @RecordCount AS INT;

```

```
SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;  
SET @QueryStart = SYSDATETIME();
```

```
SELECT  
    CONVERT(DATE,lt.TradeDateTime) AS [Date],  
    hb.BrokerID,  
    hac.AccountID,  
    hco.CompanyName,  
    hex.ExchangeID,  
    hs.SecurityCode,  
    htt.TradeTypeDesc,  
    lt.IsCashFlag,  
    lt.StatusCode,  
    lt.TradeID,  
    lt.TradeDateTime,  
    lt.Quantity,  
    lt.BidPrice,  
    lt.TradePrice,  
    lt.Charge,  
    lt.Commission,  
    lt.Tax  
INTO #temp  
FROM  
    dbo.TLNK_Trade lt (NOLOCK)  
    JOIN dbo.HUB_Broker hb (NOLOCK)  
        ON hb.HUB_Broker_SQN = lt.HUB_Broker_SQN  
    JOIN dbo.HUB_Account hac (NOLOCK)  
        ON hac.HUB_Account_SQN = lt.HUB_Account_SQN  
    JOIN dbo.HUB_Company hco (NOLOCK)  
        ON hco.HUB_Company_SQN = lt.HUB_Company_SQN  
    JOIN dbo.HUB_Exchange hex (NOLOCK)  
        ON hex.HUB_Exchange_SQN = lt.HUB_Exchange_SQN  
    JOIN dbo.HUB_Security hs (NOLOCK)  
        ON hs.HUB_Security_SQN = lt.HUB_Security_SQN  
    JOIN dbo.HUB_TradeType htt (NOLOCK)  
        ON htt.HUB_TradeType_SQN = lt.HUB_TradeType_SQN  
WHERE  
    CONVERT(DATE,lt.TradeDateTime) = '2005/01/03';
```

```

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ( '4 - Trade',@Iteration,'DV',@QueryStart,@QueryEnd,@RecordCount);

```

```

/*****
Category 5 - QuarterlyCompanyPerformance
*****/

```

```

SET NOCOUNT ON;

```

```

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

```

```

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

```

```

SELECT
    tcop.FiscalQuarter,
    tcop.FiscalYear,
    hco.CompanyName,
    tcop.FiscalQuarterStartDate,
    tcop.Revenue,
    tcop.NetEarnings,
    tcop.EarningsPerShareBasic,
    tcop.EarningsPerShareDiluted,
    tcop.Margin,
    tcop.Inventory,
    tcop.Assets,
    tcop.Liabilities,
    tcop.SharesOutstandingBasicAvg,
    tcop.SharesOutstandingDilutedAvg
INTO #temp
FROM

```

```

        dbo.TLNK_QuarterlyCompanyPerformance tcop
    JOIN dbo.HUB_Company hco
        ON hco.HUB_Company_SQN = tcop.HUB_Company_SQN
WHERE
    tcop.FiscalYear = 2004
    AND tcop.Assets >= (tcop.Liabilities * 1.5)
    AND tcop.EarningsPerShareBasic > 0.25;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ( '5 - QuarterlyCompanyPerformance', @Iteration, 'DV', @QueryStart, @QueryEnd, @RecordCount);

/*****
Category 5 - Trade
*****/

SET NOCOUNT ON;

DECLARE
    @QueryStart AS DATETIME2(7),
    @QueryEnd AS DATETIME2(7),
    @Iteration AS INT,
    @RecordCount AS INT;

SELECT @Iteration = Iteration FROM master.dbo.CurrentIteration;
SET @QueryStart = SYSDATETIME();

SELECT
    DATENAME(YEAR, lt.TradeDateTime) AS [Year],
    DATENAME(MONTH, lt.TradeDateTime) + ' ' + DATENAME(YEAR, lt.TradeDateTime) AS [Month],
    sb.BrokerName AS BrokerName,
    SUM(lt.Quantity) AS Quantity,
    SUM(lt.TradePrice * lt.Quantity) AS SalesAmount,
    SUM(lt.Commission) AS Commission
INTO #temp
FROM

```

```
    dbo.TLNK_Trade lt (NOLOCK)
  JOIN dbo.HUB_Broker hb (NOLOCK)
        ON hb.HUB_Broker_SQN = lt.HUB_Broker_SQN
  JOIN dbo.SAT_Broker sb
        ON sb.HUB_Broker_SQN = hb.HUB_Broker_SQN
WHERE
    DATEPART(YEAR,lt.TradeDateTime) = 2005
GROUP BY
    DATENAME(YEAR,lt.TradeDateTime),
    DATENAME(MONTH,lt.TradeDateTime) + ' ' + DATENAME(YEAR,lt.TradeDateTime),
    sb.BrokerName;

SET @RecordCount = @@ROWCOUNT;
SET @QueryEnd = SYSDATETIME();

INSERT INTO master.dbo.QueryResults
VALUES ('5 - Trade',@Iteration,'DV',@QueryStart,@QueryEnd,@RecordCount);
```



## 1.5 Storage Size Queries

```
USE TPCE_DM;  
GO
```

```
SELECT  
    'DV',  
    CONVERT(BIGINT,ltrim(str(@pages * 8192 / 1024.,15,0) + '')),  
    CONVERT(BIGINT,ltrim(str((@usedpages - @pages) * 8192 / 1024.,15,0) + ''));
```

```
USE TPCE_DV;  
GO
```

```
SELECT  
    'DV',  
    CONVERT(BIGINT,ltrim(str(@pages * 8192 / 1024.,15,0) + '')),  
    CONVERT(BIGINT,ltrim(str((@usedpages - @pages) * 8192 / 1024.,15,0) + ''));
```