

# **TPC BENCHMARK™ DS**

Standard Specification

Version 1.3.1

February, 2015

**Transaction Processing Performance Council (TPC)**

**[www.tpc.org](http://www.tpc.org)**

**[info@tpc.org](mailto:info@tpc.org)**

**© 2015 Transaction Processing Performance Council**

**All Rights Reserved**

## Legal Notice

The TPC reserves all right, title, and interest to this document and associated source code as provided under U.S. and international laws, including without limitation all patent and trademark rights therein.

Permission to copy without fee all or part of this document is granted provided that the TPC copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the Transaction Processing Performance Council. To copy otherwise requires specific permission.

## No Warranty

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE INFORMATION CONTAINED HEREIN IS PROVIDED “AS IS” AND WITH ALL FAULTS, AND THE AUTHORS AND DEVELOPERS OF THE WORK HEREBY DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY (IF ANY) IMPLIED WARRANTIES, DUTIES OR CONDITIONS OF MERCHANTABILITY, OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY OR COMPLETENESS OF RESPONSES, OF RESULTS, OF WORKMANLIKE EFFORT, OF LACK OF VIRUSES, AND OF LACK OF NEGLIGENCE. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THE WORK.

IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THE WORK BE LIABLE TO ANY OTHER PARTY FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THE WORK, WHETHER OR NOT SUCH AUTHOR OR DEVELOPER HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

## Trademarks

TPC Benchmark, TPC-DS, and QphDS are trademarks of the Transaction Processing Performance Council.

## Acknowledgments

Developing a TPC benchmark for a new environment requires a huge effort to conceptualize research, specify, review, prototype, and verify the benchmark. The TPC acknowledges the work and contributions of the TPC-DS subcommittee member companies in developing the TPC-DS specification.






















The TPC-DS subcommittee would like to acknowledge the contributions made by the many members during the development of the benchmark specification. It has taken the dedicated efforts of people across many companies, often in addition to their regular duties. The list of significant contributors to this version includes Susanne Englert, Mary Meredith, Sreenivas Gukal, Doug Johnson 1+2, Lubor Kollar, Murali Krishna, Bob Lane, Larry Lutz, Juergen Mueller, Bob Murphy, Doug Nelson, Ernie Ostic, Raghunath Othayoth Nambiar, Meikel Poess, Haider Rizvi, Bryan Smith, Eric Speed, Cadambi Sriram, Jack Stephens, John Susag, Tricia Thomas, Dave Walrath, Shirley Wang, Guogen Zhang, Torsten Grabs, Charles Levine, Mike Nikolaiev, Alain Crolotte, Francois Raab, Yeye He, Margaret McCarthy, Indira Patel, Daniel Pol, John Galloway, Jerry Lohr, Jerry Buggert, Michael Brey, Nicholas Wakou, Vince Carbone, Wayne Smith, Dave Steinhoff.

## Document Revision History

Date	Version	Description
01-17-2012	1.0.0	Mail ballot version
11-17-2011	1.1.0	Trickle update clarification
09-05-2014	1.2.0	Corrections for FogBugz entries: 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797
11-13-2014	1.3.0	Corrections for FogBugz entries: 779, 797, 800, 803, 804, 805, 807, 808, 811, 812, 814, 815, 816, 817, 819, 821, 822, 823, 824, 825, 826, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850
02-19-2015	1.3.1	Correction for FogBugz entries: 882,899 and 900

TPC Membership  
(as of December 2014)

Full Members

Associate Members

				
---	---	---	--	---

# Table of Contents

<b>0</b>	<b>PREAMBLE.....</b>	<b>7</b>
0.1	INTRODUCTION.....	7
0.2	GENERAL IMPLEMENTATION GUIDELINES.....	7
0.3	GENERAL MEASUREMENT GUIDELINES.....	8
0.4	WORKLOAD INDEPENDENCE .....	8
0.5	ASSOCIATED MATERIALS .....	9
<b>1</b>	<b>BUSINESS AND BENCHMARK MODEL .....</b>	<b>11</b>
1.1	OVERVIEW.....	11
1.2	BUSINESS MODEL.....	12
1.3	DATA MODEL AND DATA ACCESS ASSUMPTIONS.....	13
1.4	QUERY AND USER MODEL ASSUMPTIONS.....	13
1.5	DATA MAINTENANCE ASSUMPTIONS .....	15
<b>2</b>	<b>LOGICAL DATABASE DESIGN .....</b>	<b>17</b>
2.1	SCHEMA OVERVIEW .....	17
2.2	COLUMN DEFINITIONS.....	17
2.3	FACT TABLE DEFINITIONS.....	18
2.4	DIMENSION TABLE DEFINITIONS.....	24
2.5	IMPLEMENTATION REQUIREMENTS .....	31
2.6	DATA ACCESS TRANSPARENCY REQUIREMENTS.....	34
<b>3</b>	<b>SCALING AND DATABASE POPULATION .....</b>	<b>36</b>
3.1	SCALING MODEL .....	36
3.2	TEST DATABASE SCALING.....	36
3.3	QUALIFICATION DATABASE SCALING .....	37
3.4	DSDGEN AND DATABASE POPULATION .....	38
3.5	DATA VALIDATION.....	38
<b>4</b>	<b>QUERY OVERVIEW .....</b>	<b>39</b>
4.1	GENERAL REQUIREMENTS AND DEFINITIONS FOR QUERIES.....	39
4.2	QUERY MODIFICATION METHODS.....	40
4.3	SUBSTITUTION PARAMETER GENERATION .....	45
<b>5</b>	<b>DATA MAINTENANCE .....</b>	<b>46</b>
5.1	IMPLEMENTATION REQUIREMENTS AND DEFINITIONS .....	46
5.2	REFRESH DATA .....	47
5.3	DATA MAINTENANCE FUNCTIONS.....	49
<b>6</b>	<b>DATA PERSISTENCY PROPERTIES .....</b>	<b>69</b>
6.1	THE ACID PROPERTIES.....	69
6.2	ATOMICITY REQUIREMENTS.....	70
6.3	CONSISTENCY REQUIREMENTS.....	70
6.4	ISOLATION REQUIREMENTS.....	71
6.5	DURABILITY REQUIREMENTS .....	75
<b>7</b>	<b>PERFORMANCE METRICS AND EXECUTION RULES.....</b>	<b>78</b>
7.1	DEFINITION OF TERMS.....	78
7.2	CONFIGURATION RULES .....	78

7.3	QUERY VALIDATION .....	80
7.4	EXECUTION RULES .....	81
7.5	OUTPUT DATA.....	87
7.6	METRICS.....	87
<b>8</b>	<b>SUT AND DRIVER IMPLEMENTATION .....</b>	<b>90</b>
8.1	MODELS OF TESTED CONFIGURATIONS .....	90
8.2	SYSTEM UNDER TEST (SUT) DEFINITION .....	90
8.3	DRIVER DEFINITION .....	92
<b>9</b>	<b>PRICING .....</b>	<b>93</b>
9.1	PRICED SYSTEM.....	93
9.2	ALLOWABLE SUBSTITUTION.....	95
<b>10</b>	<b>FULL DISCLOSURE.....</b>	<b>96</b>
10.1	REPORTING REQUIREMENTS.....	96
10.2	FORMAT GUIDELINES .....	96
10.3	FULL DISCLOSURE REPORT CONTENTS .....	96
10.4	EXECUTIVE SUMMARY .....	101
10.5	AVAILABILITY OF THE FULL DISCLOSURE REPORT .....	104
10.6	REVISIONS TO THE FULL DISCLOSURE REPORT .....	104
10.7	DERIVED RESULTS .....	104
10.8	SUPPORTING FILES INDEX TABLE.....	105
10.9	SUPPORTING FILES .....	106
<b>11</b>	<b>AUDIT .....</b>	<b>108</b>
11.1	GENERAL RULES .....	108
11.2	AUDITOR'S CHECK LIST.....	108
11.3	CLAUSE 4 RELATED ITEMS.....	109
11.4	CLAUSE 5 RELATED ITEMS.....	110
11.5	CLAUSE 6 RELATED ITEMS.....	110
11.6	CLAUSE 7 RELATED ITEMS.....	110
11.7	CLAUSE 8 RELATED ITEMS.....	110
11.8	CLAUSE 9 RELATED ITEMS.....	111
11.9	CLAUSE 10 RELATED ITEMS .....	111

## 0 PREAMBLE

### 0.1 Introduction

The TPC Benchmark™ DS (TPC-DS) is a decision support benchmark that models several generally applicable aspects of a decision support system, including queries and data maintenance. The benchmark provides a representative evaluation of the System Under Test's (SUT) performance as a general purpose decision support system.

This benchmark illustrates decision support systems that:

- Examine large volumes of data;
- Give answers to real-world business questions;
- Execute queries of various operational requirements and complexities (e.g., ad-hoc, reporting, iterative OLAP, data mining);
- Are characterized by high CPU and IO load;
- Are periodically synchronized with source OLTP databases through database maintenance functions.

A benchmark result measures query throughput and data maintenance performance for a given hardware, operating system, and DBMS configuration under a controlled, complex, multi-user decision support workload.

**Comment:** While separated from the main text for readability, comments and appendices are a part of the standard and their provisions must be enforced.

### 0.2 General Implementation Guidelines

The purpose of TPC benchmarks is to provide relevant, objective performance data to industry users. To achieve that purpose, TPC benchmark specifications require benchmark tests be implemented with systems, products, technologies and pricing that:

- a) Are generally available to users;
- b) Are relevant to the market segment that the individual TPC benchmark models or represents (e.g., TPC-DS models and represents complex, high data volume, decision support environments);
- c) Would plausibly be implemented by a significant number of users in the market segment modeled or represented by the benchmark.

In keeping with these requirements, the TPC-DS database must be implemented using commercially available database management system (DBMS) software, and its queries must be executed via SQL interface.

The use of new systems, products, technologies (hardware or software) and pricing is encouraged so long as they meet the requirements above. Specifically prohibited are benchmark systems, products, technologies or pricing (hereafter referred to as "implementations") whose primary purpose is performance optimization of TPC benchmark results without any corresponding applicability to real-world applications and environments. In other words, all "benchmark special" implementations, which improve benchmark results but not real-world performance or pricing, are prohibited.

A number of characteristics shall be evaluated in order to judge whether a particular implementation is a benchmark special. It is not required that each point below be met, but that the cumulative weight of the evidence be considered to identify an unacceptable implementation. Absolute certainty or certainty beyond a reasonable doubt is not required to make a judgment on this complex issue. The question that must be answered is: "Based on the available evidence, does the clear preponderance (the greater share or weight) of evidence indicate this implementation is a benchmark special?"

The following characteristics shall be used to judge whether a particular implementation is a benchmark special:

- a) Is the implementation generally available, documented, and supported?

- b) Does the implementation have significant restrictions on its use or applicability that limits its use beyond TPC benchmarks?
- c) Is the implementation or part of the implementation poorly integrated into the larger product?
- d) Does the implementation take special advantage of the limited nature of TPC benchmarks (e.g., query templates, query mix, concurrency and/or contention, isolation requirements, etc.) in a manner that would not be generally applicable to the environment the benchmark represents?
- e) Is the use of the implementation discouraged by the vendor? (This includes failing to promote the implementation in a manner similar to other products and technologies.)
- f) Does the implementation require uncommon sophistication on the part of the end-user, programmer, or system administrator?
- g) Is the pricing unusual or non-customary for the vendor or unusual or non-customary compared to normal business practices? The following pricing practices are suspect:
  - Availability of a discount to a small subset of possible customers;
  - Discounts documented in an unusual or non-customary manner;
  - Discounts that exceed 25% on small quantities and 50% on large quantities;
  - Pricing featured as a close-out or one-time special;
  - Unusual or non-customary restrictions on transferability of product, warranty or maintenance on discounted items.
- h) Is the implementation (including beta-release components) being purchased or used for applications in the market segment the benchmark represents? How many sites implemented it? How many end-users benefit from it? If the implementation is not currently being purchased or used, is there any evidence to indicate that it will be purchased or used by a significant number of end-user sites?

### 0.3 General Measurement Guidelines

TPC benchmark results are expected to be accurate representations of system performance. Therefore, there are specific guidelines that are expected to be followed when measuring those results. The approach or methodology to be used in the measurements are either explicitly described in the specification or left to the discretion of the test sponsor.

When not described in the specification, the methodologies and approaches used must meet the following requirements:

- a) The approach is an accepted engineering practice or standard;
- b) The approach does not enhance the result;
- c) Equipment used in measuring the results is calibrated according to established quality standards;
- d) Fidelity and candor is maintained in reporting any anomalies in the results, even if not specified in the benchmark requirements.

**Comment:** The use of new methodologies and approaches is encouraged as long as they meet the requirements outlined above.

### 0.4 Workload Independence

TPC-DS uses terminology and metrics which are similar to other benchmarks originated by the TPC and others. Such similarity in terminology does not in any way imply that TPC-DS results are comparable to other benchmarks. The only benchmark results comparable to TPC-DS are other TPC-DS results compliant with the same major revision of the benchmark specification and with the same scale factor.

While this benchmark offers a rich environment representative of many decision support systems, it does not reflect the entire range of decision support requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-DS approximates the customer's application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.



Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. As a result of these and other factors, relative system performance will vary. Therefore, TPC-DS should not be used as a substitute for a specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

Benchmark sponsors are permitted to employ several possible system designs and a broad degree of implementation freedom within the constraints detailed in this specification. A full disclosure report (FDR) of the implementation details must be made available along with the reported results.

## 0.5 Associated Materials

In addition to this document, TPC-DS relies on material which is only available electronically. While not included in the printed version of the specification, this material is integral to the submission of a compliant TPC-DS benchmark result. Table 0-1 summarizes the electronic material related to the TPC-DS specification that is available for download from the TPC web site.

This material is maintained, versioned and revised independently of the specification itself. Refer to **Appendix F** to determine which version(s) of the electronic content are compliant with this revision of the specification.

**Table 0-1 Electronically Available Specification Material**

Content	File Name/Location	Usage	Additional Information
Data generator	dsdgen	Used to generate the data sets for the benchmark	Clause 3.4
Query generator	dsqgen	Used to generate the query sets for the benchmark	Clause 4.1.2
Query Templates	query_templates/	Used by <b>dsqgen</b> to generate executable query text	Clause 4.1.3
Query Template Variants	query_variants/	Used by dsqgen to generate alternative executable query text	Appendix C
Table definitions in ANSI SQL	tpcds.sql,tpcds_source.sql	Sample implementation of the logical schema for the data warehouse.	Appendix A
Data Maintenance Functions in ANSI SQL	data_maintenance/	Sample implementation of the SQL needed for the Data Maintenance phase of the benchmark	Clause 5.3
Answer Sets	answer_sets/	Used to verify the initial population of the data warehouse.	Clause 7.3
Reference Data Set	run dsdgen with -validate flag	Set of files for each scale factor to compare the correct data generation of base data, refresh data and <b>dsqgen</b> data	

0.5.1 The rules for pricing are included in the current revision of the TPC Pricing Specification located on the TPC website (<http://www.tpc.org>).

**Comment:** There is a non-binding HowTo.doc guide electronically available. The purpose of this guide is to describe the most common tasks necessary to implement a TPC-DS benchmark. The target audience is individuals who want to install, populate, run and analyze the database, queries and data maintenance workloads for TPC-DS.

# **1 Business and Benchmark Model**

## **1.1 Overview**

TPC Benchmark™ DS (TPC-DS) contains benchmark components that can be used to assess a broad range of system topologies and implementation methodologies in a technically rigorous and directly comparable, vendor-neutral manner. In order to ease the learning curve for users and benchmark sponsors who are new to TPC-DS, the benchmark has been mapped to a typical business environment. This clause outlines the business modeling assumptions that were adopted during the development of the benchmark, and their impact on the benchmark environment.

TPC-DS models the decision support functions of a retail product supplier. The supporting schema contains vital business information, such as customer, order, and product data. The benchmark models the two most important components of any mature decision support system:

- User queries, which convert operational facts into business intelligence.
- Data maintenance, which synchronizes the process of management analysis with the operational external data source on which it relies.

The combination characterizes the core competency of any decision support system. TPC-DS provides a robust, rigorous and complete means for the evaluation of systems meant to provide that competency.

The benchmark abstracts the diversity of operations found in an information analysis application, while retaining essential performance characteristics. As it is necessary to execute a great number of queries and data transformations to completely manage any business analysis environment, no benchmark can succeed in exactly mimicking a particular environment and remain broadly applicable.

While TPC-DS does not aspire to be a model of how to build an actual information analysis application, the workload has been granted a realistic context. It imitates the activity of a multi-channel retailer; thus tracking store, web and catalog sales channels.

The goal of selecting a retail business model is to assist the reader in relating intuitively to the components of the benchmark, without tracking that industry segment so tightly as to minimize the relevance of the benchmark. The TPC-DS workload may be used to characterize any industry that must transform operational and external data into business intelligence.

Although the emphasis is on information analysis, the benchmark recognizes the need to periodically refresh the database. The database is not a one-time snapshot of a business operations database nor is it a database where OLTP applications are running concurrently.

Some TPC benchmarks model the operational aspect of the business environment where transactions are executed on a real time basis. Other benchmarks address the simpler, more static model of decision support. The TPC-DS benchmark, models the challenges of business intelligence systems where operational data is used both to support the making of sound business decisions in near real time and to direct long-range planning and exploration.

**Figure 1-1** illustrates TPC-DS benchmark components.

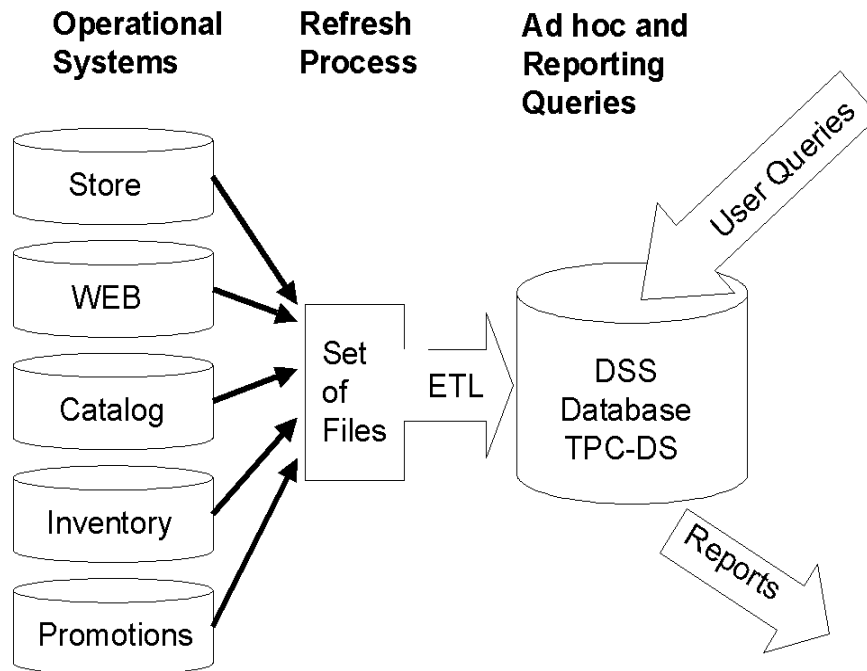


Figure 1-1: TPC-DS benchmark components

## 1.2 Business Model

TPC-DS models any industry that must manage, sell and distribute products (e.g., food, electronics, furniture, music and toys etc.). It utilizes the business model of a large retail company having multiple stores located nation-wide. Beyond its brick and mortar stores, the company also sells goods through catalogs and the Internet. Along with tables to model the associated sales and returns, it includes a simple inventory system and a promotion system.

The following are examples of business processes of this retail company:

- Record customer purchases (and track customer returns) from any sales channel
- Modify prices according to promotions
- Maintain warehouse inventory
- Create dynamic web pages
- Maintain customer profiles (Customer Relationship Management)

TPC-DS does not benchmark the operational systems. It is assumed that the channel sub-systems were designed at different times by diverse groups having dissimilar functional requirements. It is also recognized that they may be operating on significantly different hardware configurations, software configurations and data model semantics. All three channel sub-systems are autonomous and retain possibly redundant information regarding customers, addresses, etc. For more information in the benchmarking of operational system, please see the TPC website (<http://www.tpc.org>).

TPC-DS' modeling of the business environment falls into three broad categories:

- Data Model and Data Access Assumptions (see Clause 1.3)

- Query and User Model Assumptions (see Clause 1.4)
- Data Maintenance Assumptions(see Clause 1.5)

### 1.3 Data Model and Data Access Assumptions

- 1.3.1 The TPC-DS models a database that is continuously available 24 hours a day, 7 days a week, for data modifications against any/all tables and various types (e.g. ad hoc, reporting, iterative OLAP, data mining) of queries originating from multiple concurrent end user/class sessions. This environment allows potentially long running and multi-part queries where the DBA cannot assume that the database can be quiescent during any particular period. In addition, this mix of queries and data maintenance functions is subject to specific ACID requirements, since queries and data maintenance functions may execute concurrently. The same ACID requirements are mandatory for all query classes.
- 1.3.2 The TPC-DS database tracks, possibly with some delay, the data state of an operational database through data maintenance functions, which include a number of modifications impacting some part of the decision support database.
- 1.3.3 The TPC-DS schema is a snowflake schema. It consists of multiple dimension and fact tables. Each dimension has a single column surrogate key. The fact tables join with dimensions using each dimension table's surrogate key. The dimension tables can be classified into one of the following types:
- **Static:** The contents of the dimension are loaded once during database load and do not change over time. The date dimension is an example of a static dimension.
  - **Historical:** The history of the changes made to the dimension data is maintained by creating multiple rows for a single business key value. Each row includes columns indicating the time period for which the row is valid. The fact tables are linked to the dimension values that were active at the time the fact was recorded, thus maintaining “historical truth”. Item is an example of a historical dimension.
  - **Non-Historical:** The history of the changes made to the dimension data is not maintained. As dimension rows are updated, the previous values are overwritten and this information is lost. All fact data is associated with the most current value of the dimension. Customer is an example of a Non-Historical dimension.
- 1.3.4 To achieve the optimal compromise between performance and operational consistency, the database administrator can set, once and for all, the locking levels and the concurrent scheduling rules for queries and data maintenance functions.
- 1.3.5 The size of a DSS system – more precisely the size of the data captured in a DSS system – may vary from company to company and within the same company based on different time frames. Therefore, the TPC-DS benchmark will model several different sizes of the DSS (a.k.a. benchmark scaling or scale factor).

### 1.4 Query and User Model Assumptions

The users and queries modeled by the benchmark exhibit the following characteristics:

- a) They address complex business problems
- b) They use a variety of access patterns, query phrasings, operators, and answer set constraints
- c) They employ query parameters that change across query executions

In order to address the enormous range of query types and user behaviors encountered by a decision support system, TPC-DS utilizes a generalized query model. This model allows the benchmark to capture important aspects of the interactive, iterative nature of on-line analytical processing (OLAP) queries, the longer-running complex queries of data mining and knowledge discovery, and the more planned behavior of well known report queries.

#### 1.4.1 Query Classes

The size of the schema and its three sales channels allow for amalgamating the above query classes, especially ad-hoc and reporting, into the same benchmark. An ad-hoc querying workload simulates an environment in which users connected to the database system send individual queries that are not known in advance. The system's database administrator (DBA) cannot optimize the database system specifically for this set of queries. Consequently, execution time for those queries can be very long. In contrast, queries in a reporting workload are very well known in advance. As a result, the DBA can optimize the database system specifically for these queries to execute them very rapidly by using clever data placement methods (e.g. partitioning and clustering) and auxiliary data structures (e.g. materialized views and indexes). Amalgamating both types of queries has been traditionally difficult in benchmark environments since per the definition of a benchmark all queries, apart from bind variables, are known in advance. TPC-DS accomplishes this fusion by dividing the schema into reporting and ad-hoc parts. The catalog sales channel is dedicated for the reporting part, while the store and web channels are dedicated for the ad-hoc part. The catalog sales channel was chosen as the reporting part because its data accounts for 40% of the entire database. For the reporting part of the schema complex auxiliary data structures are allowed, while for the ad-hoc part only basic auxiliary data structures are allowed. The idea behind this approach is that the queries accessing the ad-hoc part constitute the ad-hoc query set while the queries accessing the reporting part are considered the reporting queries.

A sophisticated decision support system must support a diverse user population. While there are many ways to categorize those diverse users and the queries that they generate, TPC-DS has defined four broad classes of queries that characterize most decision support queries:

- Reporting queries
- Ad hoc queries
- Iterative OLAP queries
- Data mining queries

TPC-DS provides a wide variety of queries in the benchmark to emulate these diverse query classes.

#### 1.4.1.1 Reporting Queries

These queries capture the “reporting” nature of a DSS system. They include queries that are executed periodically to answer well-known, pre-defined questions about the financial and operational health of a business. Although reporting queries tend to be static, minor changes are common. From one use of a given reporting query to the next, a user might choose to shift focus by varying a date range, geographic location or a brand name.

#### 1.4.1.2 Ad hoc Queries

These queries capture the dynamic nature of a DSS system in which impromptu queries are constructed to answer immediate and specific business questions. The central difference between *ad hoc* queries and reporting queries is the limited degree of foreknowledge that is available to the DBA when planning for an *ad hoc* query.

#### 1.4.1.3 Iterative OLAP Queries

OLAP queries allow for the exploration and analysis of business data to discover new and meaningful relationships and trends. While this class of queries is similar to the “Ad hoc Queries” class, it is distinguished by a scenario-based user session in which a sequence of queries is submitted. Such a sequence may include both complex and simple queries.

#### 1.4.1.4 Data Mining Queries

Data mining is the process of sifting through large amounts of data to produce data content relationships. It can predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. This class of queries typically consists of joins and large aggregations that return large data result sets for possible extraction.

## 1.5 Data Maintenance Assumptions

A data warehouse is only as accurate and current as the operational data on which it is based. Accordingly, the migration of data from operational OLTP systems to analytical DSS systems is crucial. The migration tends to vary widely from business to business and application to application. Previous benchmarks evaluated the data analysis component of decision support systems while excluding a realistic data refresh process. TPC-DS offers a more balanced view.

Decision support database refresh processes usually involve three distinct and important steps:

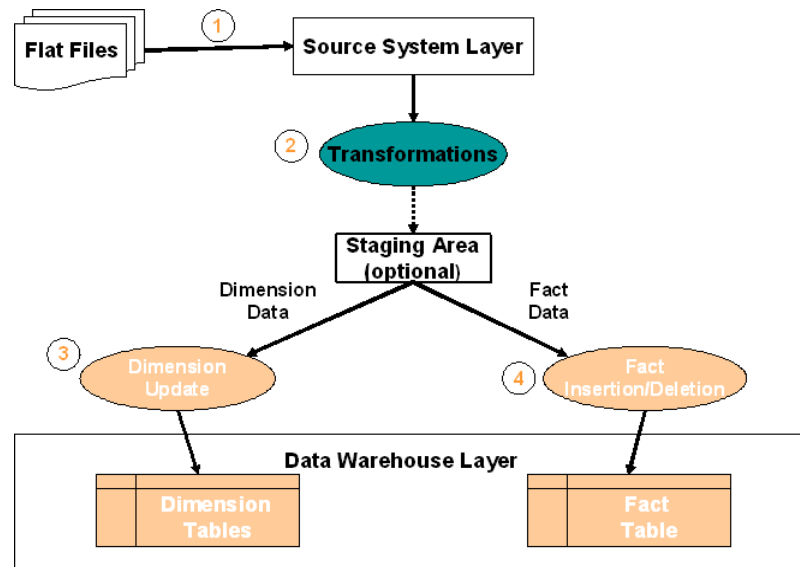
- **Data Extraction:** This phase consists of the accurate extraction of pertinent data from production OLTP databases and other relevant data sources. In a production environment, the extraction step may include numerous separate extract operations executed against multiple OLTP databases and auxiliary data sources. While selection and tuning of the associated systems and procedures is important to the success of the production system, it is separate from the purchase and configuration of the decision support servers. Accordingly, the data extract step of the ETL process (E) is not modeled in the benchmark. The TPC-DS data maintenance process starts from generated flat files that are assumed to be the output of this external Extraction process.
- **Data Transformation:** This is when the extracted data is cleansed and massaged into a common format suitable for assimilation by the decision support database.
- **Data Load:** This is the actual insertion, modification and deletion of data within the decision support database.

Taken together, the progression of Extraction, Transformation and Load is more commonly known by its acronym, ETL. In TPC-DS, the modeling of Transformation and Load is known as Data Maintenance (DM) or Data Refresh. In this specification the two terms are used interchangeably.

The DM process of TPC-DS includes the following tasks that result from such a complex business environment as shown in Figure 1-2:

- i) Load the refresh data set, which consists of new, deleted and changed data destined for the data warehouse in its operational format.
- ii) Load refresh data set into the data warehouse applying data transformations, e.g.:
  - Data denormalization (3rd Normal form to snowflake). During this step the source tables are mapped into the data warehouse by:
    - Direct source to target mapping. This type of mapping is the most common. It applies to tables in the data warehouse that have an equivalent table in the operational schema.
    - Multiple data warehouse source tables are joined and the result is mapped to one target table. This mapping translates the third normal form of the operational schema into the de-normalized form of the data warehouse.
    - One source table is mapped to multiple target tables. This mapping is the least common. It occurs if, for efficiency reason, the schema of the operational system is less normalized than the data warehouse schema.
  - Syntactically cleanse data
  - De-normalize
- iii) Properly manage data that is subject to version control and historical retention (i.e., slowly changing dimensions)
- iv) Insert new fact records and delete fact records by date.

The structure and relationships between the flat files is provided in form of a table description and the ddl of the tables that represent the hypothetical operational database in Appendix A.



**Figure 1-2: Execution Overview of the Data Maintenance Process**

Each flat file contains new and changed rows. It is assumed that fact data is not changed in the operational system. In order to find out whether a row is new or changed it needs to be mapped to the surrogate key table. If the ODS key for this row exists in the surrogate key table, it can be assumed that the row is changed. Otherwise the row is new.



## 2 Logical Database Design

### 2.1 Schema Overview

The TPC-DS schema models the sales and sales returns process for an organization that employs three primary sales channels: stores, catalogs, and the Internet. The schema includes seven fact tables:

- A pair of fact tables focused on the product sales and returns for each of the three channels
- A single fact table that models inventory for the catalog and internet sales channels.

In addition, the schema includes 17 dimension tables that are associated with all sales channels. The following clauses specify the logical design of each table:

- The name of the table, along with its abbreviation (listed parenthetically)
- A logical diagram of each fact table and its related dimension tables
- The high-level definitions for each table and its relationship to other tables, using the format defined in Clause 2.2
- The scaling and cardinality information for each column

### 2.2 Column Definitions

#### 2.2.1 Column Name

2.2.1.1 Each column is uniquely named, and each column name begins with the abbreviation of the table in which it appears.

2.2.1.2 Columns that are part of the table's primary key are indicated in the column called Primary Key (Sections 2.3 and 2.4). If a table uses a composite primary key, then for convenience of reading the order of a given column in a table's primary key is listed in parentheses following the column name.

2.2.1.3 Columns that are part of a business key are indicated with (B) appearing after the column name (Sections 2.3 and 2.4). A business key is neither a primary key nor a foreign key in the context of the data warehouse schema. It is only used to differentiate new data from update data of the source tables during the data maintenance operations.

#### 2.2.2 Datatype

2.2.2.1 Each column employs one of the following datatypes:

- a) Identifier means that the column shall be able to hold any key value generated for that column.
- b) Integer means that the column shall be able to exactly represent integer values (i.e., values in increments of 1) in the range of at least  $(-2^{n-1})$  to  $(2^{n-1} - 1)$ , where  $n$  is 64.
- c) Decimal( $d, f$ ) means that the column shall be able to represent decimal values up to and including  $d$  digits, of which  $f$  shall occur to the right of the decimal place; the values can be either represented exactly or interpreted to be in this range.
- d) Char( $N$ ) means that the column shall be able to hold any string of characters of a fixed length of  $N$ .

**Comment:** If the string that a column of datatype char( $N$ ) holds is shorter than  $N$  characters, then trailing spaces shall be stored in the database or the database shall automatically pad with spaces upon retrieval such that a CHAR\_LENGTH() function will return  $N$ .

- e) Varchar( $N$ ) means that the column shall be able to hold any string of characters of a variable length with a maximum length of  $N$ . Columns defined as "varchar( $N$ )" may optionally be implemented as "char( $N$ )".
- f) Date means that the column shall be able to express any calendar day between January 1, 1900 and December 31, 2199.

2.2.2.2 The datatypes do not correspond to any specific SQL-standard datatype. The definitions are provided to highlight the properties that are required for a particular column. The benchmark implementer may employ any internal representation or SQL datatype that meets those requirements.

2.2.2.3 The implementation chosen by the test sponsor for a particular datatype definition shall be applied consistently to all the instances of that datatype definition in the schema, except for identifier columns, whose datatype may be selected to satisfy database scaling requirements.

## 2.2.3 NULLs

If a column definition includes an 'N' in the **NULLs** column, an implementer may assume that this column is populated in every row of the table for all scale factors. If the field is blank, an implementer shall assume that this column may contain NULLs.

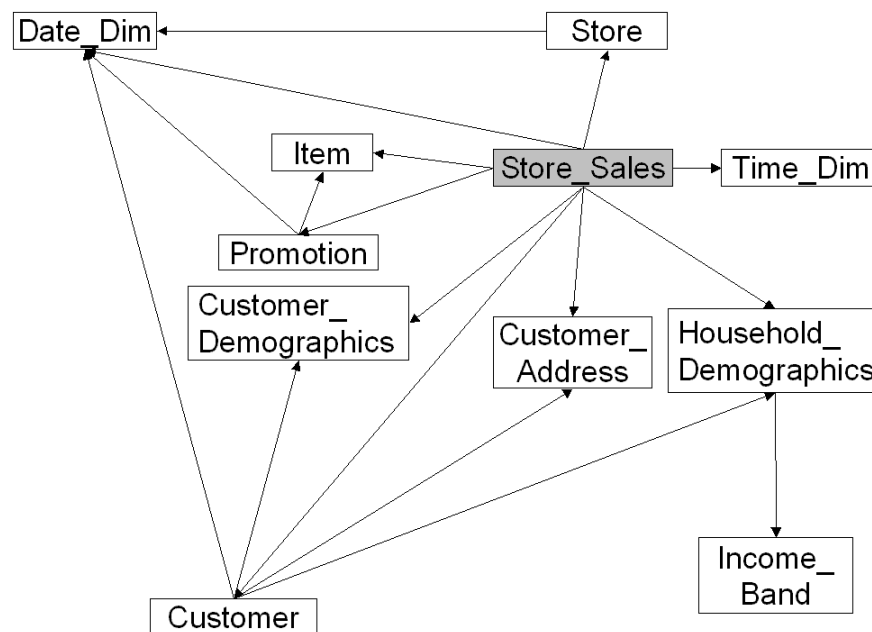
## 2.2.4 Foreign Key

If the values in this column join with another column, the foreign columns name is listed in the **Foreign Key** field of the column definition.

## 2.3 Fact Table Definitions

### 2.3.1 Store Sales (SS)

#### 2.3.1.1 Store Sales ER-Diagram



#### 2.3.1.2 Store Sales Column Definitions

Each row in this table represents a single lineitem for a sale made through the store channel and recorded in the store\_sales fact table.

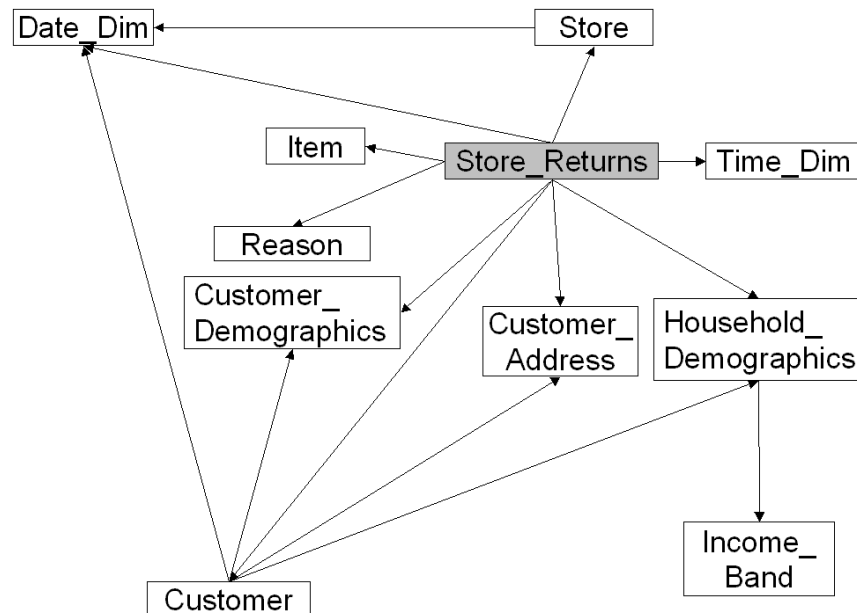
**Table 2-1 Store\_sales Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
ss_sold_date_sk	identifier			d_date_sk
ss_sold_time_sk	identifier			t_time_sk
ss_item_sk (1)	identifier	N	Y	i_item_sk
ss_customer_sk	identifier			c_customer_sk
ss_cdemo_sk	identifier			cd_demo_sk
ss_hdemo_sk	identifier			hd_demo_sk
ss_addr_sk	identifier			ca_address_sk

Column	Datatype	NULLs	Primary Key	Foreign Key
ss_store_sk	identifier			s_store_sk
ss_promo_sk	identifier			p_promo_sk
ss_ticket_number (2)	identifier	N	Y	
ss_quantity	integer			
ss_wholesale_cost	decimal(7,2)			
ss_list_price	decimal(7,2)			
ss_sales_price	decimal(7,2)			
ss_ext_discount_amt	decimal(7,2)			
ss_ext_sales_price	decimal(7,2)			
ss_ext_wholesale_cost	decimal(7,2)			
ss_ext_list_price	decimal(7,2)			
ss_ext_tax	decimal(7,2)			
ss_coupon_amt	decimal(7,2)			
ss_net_paid	decimal(7,2)			
ss_net_paid_inc_tax	decimal(7,2)			
ss_net_profit	decimal(7,2)			

### 2.3.2 Store Returns (SR)

#### 2.3.2.1 Store Returns ER-Diagram



#### 2.3.2.2 Store Returns Column Definition

Each row in this table represents a single lineitem for the return of an item sold through the store channel and recorded in the store\_returns fact table.

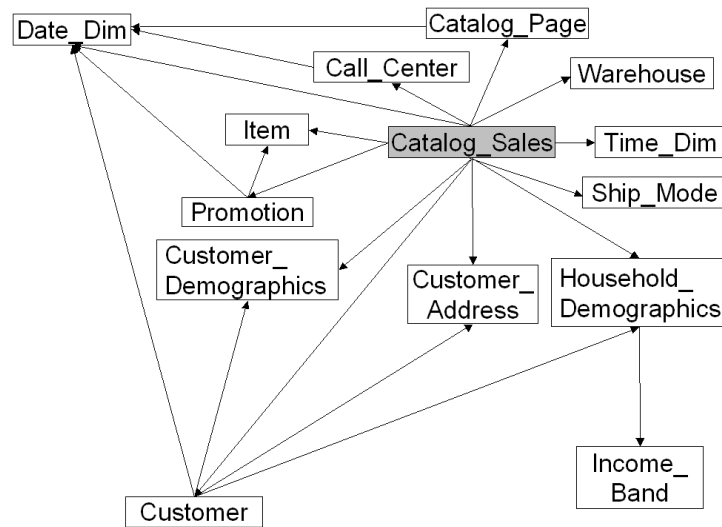
**Table 2-2 Store\_returns Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
sr_returned_date_sk	identifier			d_date_sk
sr_return_time_sk	identifier			t_time_sk
sr_item_sk (1)	identifier	N	Y	i_item_sk,ss_item_sk
sr_customer_sk	identifier			c_customer_sk
sr_demo_sk	identifier			cd_demo_sk
sr_hdemo_sk	identifier			hd_demo_sk
sr_addr_sk	identifier			ca_address_sk
sr_store_sk	identifier			s_store_sk
sr_reason_sk	identifier			r_reason_sk
sr_ticket_number (2)	identifier	N	Y	ss_ticket_number

Column	Datatype	NULLs	Primary Key	Foreign Key
sr_return_quantity	integer			
sr_return_amt	decimal(7,2)			
sr_return_tax	decimal(7,2)			
sr_return_amt_inc_tax	decimal(7,2)			
sr_fee	decimal(7,2)			
sr_return_ship_cost	decimal(7,2)			
sr_refunded_cash	decimal(7,2)			
sr_reversed_charge	decimal(7,2)			
sr_store_credit	decimal(7,2)			
sr_net_loss	decimal(7,2)			

### 2.3.3 Catalog Sales (CS)

#### 2.3.3.1 Catalog Sales ER-Diagram



#### 2.3.3.2 Catalog Sales Column Definition

Each row in this table represents a single lineitem for a sale made through the catalog channel and recorded in the catalog\_sales fact table.

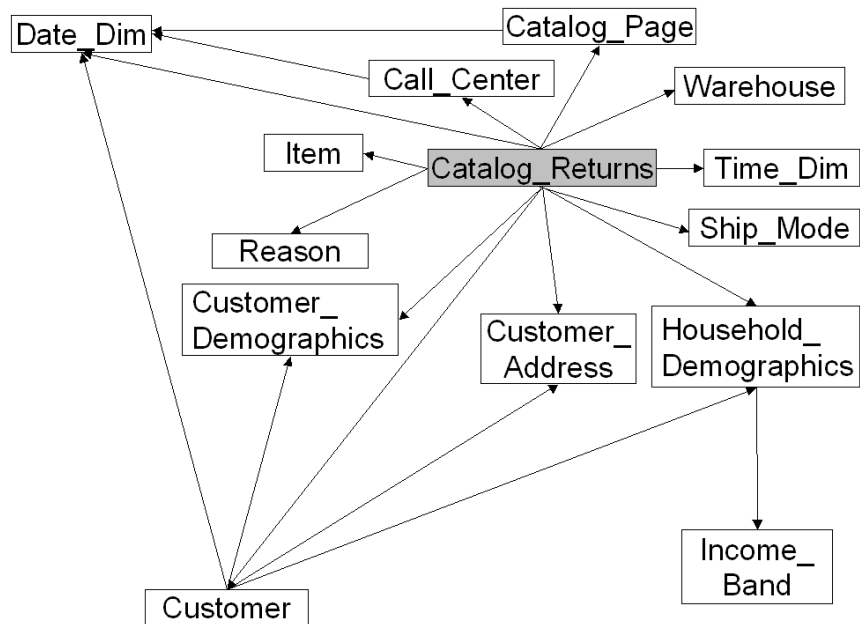
**Table 2-3 Catalog Sales Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
cs_sold_date_sk	identifier			d_date_sk
cs_sold_time_sk	identifier			t_time_sk
cs_ship_date_sk	identifier			d_date_sk
cs_bill_customer_sk	identifier			c_customer_sk
cs_bill_cdemo_sk	identifier			cd_demo_sk
cs_bill_hdemo_sk	identifier			hd_demo_sk
cs_bill_addr_sk	identifier			ca_address_sk
cs_ship_customer_sk	identifier			c_customer_sk
cs_ship_cdemo_sk	identifier			cd_demo_sk
cs_ship_hdemo_sk	identifier			hd_demo_sk
cs_ship_addr_sk	identifier			ca_address_sk
cs_call_center_sk	identifier			cc_call_center_sk
cs_catalog_page_sk	identifier			cp_catalog_page_sk
cs_ship_mode_sk	identifier			sm_ship_mode_sk
cs_warehouse_sk	identifier			w_warehouse_sk
cs_item_sk (1)	identifier	N	Y	i_item_sk
cs_promo_sk	identifier			p_promo_sk
cs_order_number (2)	identifier	N	Y	
cs_quantity	integer			

Column	Datatype	NULLs	Primary Key	Foreign Key
cs_wholesale_cost	decimal(7,2)			
cs_list_price	decimal(7,2)			
cs_sales_price	decimal(7,2)			
cs_ext_discount_amt	decimal(7,2)			
cs_ext_sales_price	decimal(7,2)			
cs_ext_wholesale_cost	decimal(7,2)			
cs_ext_list_price	decimal(7,2)			
cs_ext_tax	decimal(7,2)			
cs_coupon_amt	decimal(7,2)			
cs_ext_ship_cost	decimal(7,2)			
cs_net_paid	decimal(7,2)			
cs_net_paid_inc_tax	decimal(7,2)			
cs_net_paid_inc_ship	decimal(7,2)			
cs_net_paid_inc_ship_tax	decimal(7,2)			
cs_net_profit	decimal(7,2)			

### 2.3.4 Catalog Returns (CR)

#### 2.3.4.1 Catalog Returns ER-Diagram



#### 2.3.4.2 Catalog Returns Column Definition

Each row in this table represents a single lineitem for the return of an item sold through the catalog channel and recorded in the catalog\_returns table.

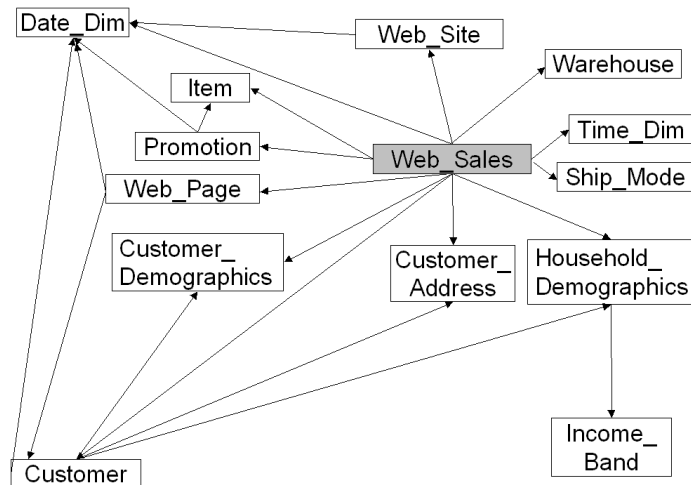
**Table 2-4 Catalog\_returns Column Definition**

Colum	Datatype	NULLs	Primary Key	Foreign Key
cr_returned_date_sk	identifier			d_date_sk
cr_returned_time_sk	identifier			t_time_sk
cr_item_sk (1)	identifier	N	Y	i_item_sk,cs_item_sk
cr_refunded_customer_sk	identifier			c_customer_sk
cr_refunded_demo_sk	identifier			cd_demo_sk
cr_refunded_hdemo_sk	identifier			hd_demo_sk
cr_refunded_addr_sk	identifier			ca_address_sk
cr_returning_customer_sk	identifier			c_customer_sk
cr_returning_demo_sk	identifier			cd_demo_sk
cr_returning_hdemo_sk	identifier			hd_demo_sk

Column	Datatype	NULLs	Primary Key	Foreign Key
cr_returning_addr_sk	identifier			ca_address_sk
cr_call_center_sk	identifier			cc_call_center_sk
cr_catalog_page_sk	identifier			cp_catalog_page_sk
cr_ship_mode_sk	identifier			sm_ship_mode_sk
cr_warehouse_sk	identifier			w_warehouse_sk
cr_reason_sk	identifier			r_reason_sk
cr_order_number (2)	identifier	N	Y	cs_order_number
cr_return_quantity	integer			
cr_return_amount	decimal(7,2)			
cr_return_tax	decimal(7,2)			
cr_return_amt_inc_tax	decimal(7,2)			
cr_fee	decimal(7,2)			
cr_return_ship_cost	decimal(7,2)			
cr_refunded_cash	decimal(7,2)			
cr_reversed_charge	decimal(7,2)			
cr_store_credit	decimal(7,2)			
cr_net_loss	decimal(7,2)			

### 2.3.5 Web Sales (WS)

#### 2.3.5.1 Web Sales ER-Diagram



#### 2.3.5.2 Web Sales Column Definition

Each row in this table represents a single lineitem for a sale made through the web channel and recorded in the web\_sales fact table.

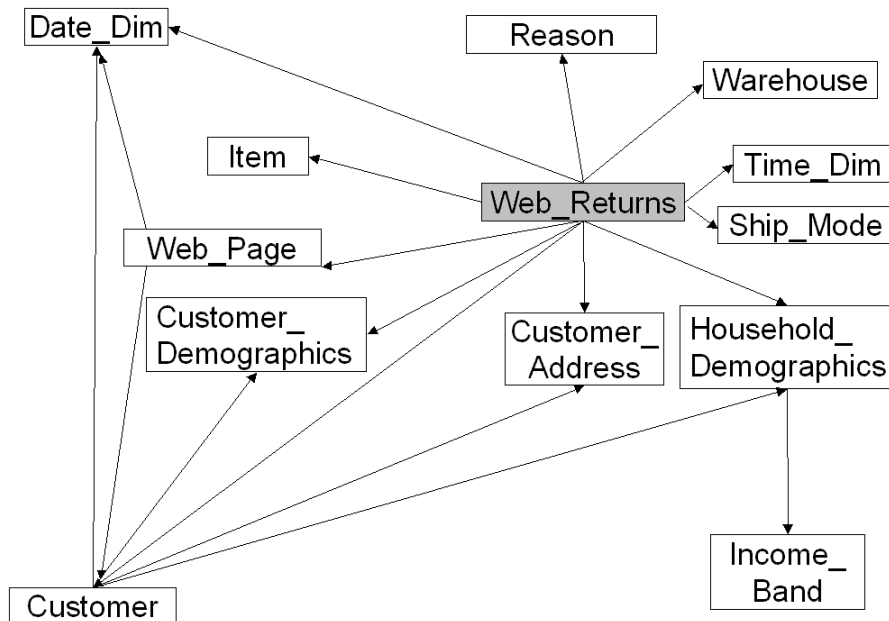
**Table 2-5 Web\_sales Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
ws_sold_date_sk	identifier			d_date_sk
ws_sold_time_sk	identifier			t_time_sk
ws_ship_date_sk	identifier			d_date_sk
ws_item_sk (1)	identifier	N	Y	i_item_sk
ws_bill_customer_sk	identifier			c_customer_sk
ws_bill_cdemo_sk	identifier			cd_demo_sk
ws_bill_hdemo_sk	identifier			hd_demo_sk
ws_bill_addr_sk	identifier			ca_address_sk
ws_ship_customer_sk	identifier			c_customer_sk
ws_ship_cdemo_sk	identifier			cd_demo_sk
ws_ship_hdemo_sk	identifier			hd_demo_sk
ws_ship_addr_sk	identifier			ca_address_sk
ws_web_page_sk	identifier			wp_web_page_sk

Column	Datatype	NULLs	Primary Key	Foreign Key
ws_web_site_sk	identifier			web_site_sk
ws_ship_mode_sk	identifier			sm_ship_mode_sk
ws_warehouse_sk	identifier			w_warehouse_sk
ws_promo_sk	identifier			p_promo_sk
ws_order_number (2)	identifier	N	Y	
ws_quantity	integer			
ws_warehouse_cost	decimal(7,2)			
ws_list_price	decimal(7,2)			
ws_sales_price	decimal(7,2)			
ws_ext_discount_amt	decimal(7,2)			
ws_ext_sales_price	decimal(7,2)			
ws_ext_warehouse_cost	decimal(7,2)			
ws_ext_list_price	decimal(7,2)			
ws_ext_tax	decimal(7,2)			
ws_coupon_amt	decimal(7,2)			
ws_ext_ship_cost	decimal(7,2)			
ws_net_paid	decimal(7,2)			
ws_net_paid_inc_tax	decimal(7,2)			
ws_net_paid_inc_ship	decimal(7,2)			
ws_net_paid_inc_ship_tax	decimal(7,2)			
ws_net_profit	decimal(7,2)			

### 2.3.6 Web Returns (WR)

#### 2.3.6.1 Web Returns ER-Diagram



#### 2.3.6.2 Web Returns Column Definition

Each row in this table represents a single lineitem for the return of an item sold through the web sales channel and recorded in the web\_returns table.

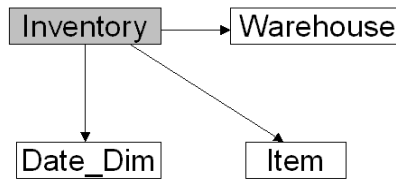
**Table 2-6 Web\_returns Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
wr_returned_date_sk	identifier			d_date_sk
wr_returned_time_sk	identifier			t_time_sk
wr_item_sk (2)	identifier	N	Y	i_item_sk,ws_item_sk
wr_refunded_customer_sk	identifier			c_customer_sk
wr_refunded_demo_sk	identifier			cd_demo_sk

Column	Datatype	NULLs	Primary Key	Foreign Key
wr_refunded_hdemo_sk	identifier			hd_demo_sk
wr_refunded_addr_sk	identifier			ca_address_sk
wr_returning_customer_sk	identifier			c_customer_sk
wr_returning_cdemo_sk	identifier			cd_demo_sk
wr_returning_hdemo_sk	identifier			hd_demo_sk
wr_returning_addr_sk	identifier			ca_address_sk
wr_web_page_sk	identifier			wp_web_page_sk
wr_reason_sk	identifier			r_reason_sk
wr_order_number (1)	identifier	N	Y	ws_order_number
wr_return_quantity	integer			
wr_return_amt	decimal(7,2)			
wr_return_tax	decimal(7,2)			
wr_return_amt_inc_tax	decimal(7,2)			
wr_fee	decimal(7,2)			
wr_return_ship_cost	decimal(7,2)			
wr_refunded_cash	decimal(7,2)			
wr_reversed_charge	decimal(7,2)			
wr_account_credit	decimal(7,2)			
wr_net_loss	decimal(7,2)			

### 2.3.7 Inventory (INV)

#### 2.3.7.1 Inventory ER-Diagram



#### 2.3.7.2 Inventory Column Definition

Each row in this table represents the quantity of a particular item on-hand at a given warehouse during a specific week.

Table 2-7 Inventory Column Definitions

Column	Datatype	NULLs	Primary Key	Foreign Key
inv_date_sk (1)	identifier	N	Y	d_date_sk
inv_item_sk (2)	identifier	N	Y	i_item_sk
inv_warehouse_sk (3)	identifier	N	Y	w_warehouse_sk
inv_quantity_on_hand	integer			

## 2.4 Dimension Table Definitions

### 2.4.1 Store (S)

Each row in this dimension table represents details of a store.

Table 2-8: Store Column Definitions

Column	Datatype	NULLs	Primary Key	Foreign Key
s_store_sk	identifier	N	Y	
s_store_id (B)	char(16)	N		



Column	Datatype	NULLs	Primary Key	Foreign Key
s_rec_start_date	date			
s_rec_end_date	date			
s_closed_date_sk	identifier			d_date_sk
s_store_name	varchar(50)			
s_number_employees	integer			
s_floor_space	integer			
s_hours	char(20)			
S_manager	varchar(40)			
S_market_id	integer			
S_geography_class	varchar(100)			
S_market_desc	varchar(100)			
s_market_manager	varchar(40)			
s_division_id	integer			
s_division_name	varchar(50)			
s_company_id	integer			
s_company_name	varchar(50)			
s_street_number	varchar(10)			
s_street_name	varchar(60)			
s_street_type	char(15)			
s_suite_number	char(10)			
s_city	varchar(60)			
s_county	varchar(30)			
s_state	char(2)			
s_zip	char(10)			
s_country	varchar(20)			
s_gmt_offset	decimal(5,2)			
s_tax_percentage	decimal(5,2)			

#### 2.4.2 Call Center (CC)

Each row in this table represents details of a call center.

**Table 2-9 Call\_center Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
cc_call_center_sk	identifier	N	Y	
cc_call_center_id (B)	char(16)	N		
cc_rec_start_date	date			
cc_rec_end_date	date			
cc_closed_date_sk	integer			d_date_sk
cc_open_date_sk	integer			d_date_sk
cc_name	varchar(50)			
cc_class	varchar(50)			
cc_employees	integer			
cc_sq_ft	integer			
cc_hours	char(20)			
cc_manager	varchar(40)			
cc_mkt_id	integer			
cc_mkt_class	char(50)			
cc_mkt_desc	varchar(100)			
cc_market_manager	varchar(40)			
cc_division	integer			
cc_division_name	varchar(50)			
cc_company	integer			
cc_company_name	char(50)			
cc_street_number	char(10)			
cc_street_name	varchar(60)			
cc_street_type	char(15)			
cc_suite_number	char(10)			
cc_city	varchar(60)			
cc_county	varchar(30)			
cc_state	char(2)			
cc_zip	char(10)			
cc_country	varchar(20)			

Column	Datatype	NULLs	Primary Key	Foreign Key
cc_gmt_offset	decimal(5,2)			
cc_tax_percentage	decimal(5,2)			

#### 2.4.3 Catalog\_page (CP)

Each row in this table represents details of a catalog page.

**Table 2-10 Catalog Page Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
<b>cp_catalog_page_sk</b>	identifier	N	Y	
cp_catalog_page_id (B)	char(16)	N		
cp_start_date_sk	integer			d_date_sk
cp_end_date_sk	integer,			d_date_sk
cp_department	varchar(50)			
cp_catalog_number	integer,			
cp_catalog_page_number	integer,			
cp_description	varchar(100)			
cp_type	varchar(100)			

#### 2.4.4 Web\_site (WEB)

Each row in this table represents details of a web site.

**Table 2-11 Web\_site Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
<b>web_site_sk</b>	identifier	N	Y	
web_site_id (B)	char(16)	N		
web_rec_start_date	date			
web_rec_end_date	date			
web_name	varchar(50)			
web_open_date_sk	identifier			d_date_sk
web_close_date_sk	identifier			d_date_sk
web_class	varchar(50)			
web_manager	varchar(40)			
web_mkt_id	integer			
web_mkt_class	varchar(50)			
web_mkt_desc	varchar(100)			
web_market_manager	varchar(40)			
web_company_id	integer			
web_company_name	char(50)			
web_street_number	char(10)			
web_street_name	varchar(60)			
web_street_type	char(15)			
web_suite_number	char(10)			
web_city	varchar(60)			
web_county	varchar(30)			
web_state	char(2)			
web_zip	char(10)			
web_country	varchar(20)			
web_gmt_offset	decimal(5,2)			
web_tax_percentage	decimal(5,2)			

#### 2.4.5 Web\_page (WP)

Each row in this table represents details of a web page within a web site.

**Table 2-12 Web\_page Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
<b>wp_web_page_sk</b>	identifier	N	Y	
wp_web_page_id (B)	char(16)	N		
wp_rec_start_date	date			

Column	Datatype	NULLs	Primary Key	Foreign Key
wp_rec_end_date	date			
wp_creation_date_sk	identifier			d_date_sk
wp_access_date_sk	identifier			d_date_sk
wp_autogen_flag	char(1)			
wp_customer_sk	identifier			c_customer_sk
wp_url	varchar(100)			
wp_type	char(50)			
wp_char_count	integer			
wp_link_count	integer			
wp_image_count	integer			
wp_max_ad_count	integer			

#### 2.4.6 Warehouse (W)

Each row in this dimension table represents a warehouse where items are stocked.

**Table 2-13 Warehouse Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
w_warehouse_sk	identifier	N	Y	
w_warehouse_id (B)	char(16)	N		
w_warehouse_name	varchar(20)			
w_warehouse_sq_ft	integer			
w_street_number	char(10)			
w_street_name	varchar(60)			
w_street_type	char(15)			
w_suite_number	char(10)			
w_city	varchar(60)			
w_county	varchar(30)			
w_state	char(2)			
w_zip	char(10)			
w_country	varchar(20)			
w_gmt_offset	decimal(5,2)			

#### 2.4.7 Customer (C)

Each row in this dimension table represents a customer.

**Table 2-14: Customer Table Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
c_customer_sk	identifier	N	Y	
c_customer_id (B)	char(16)	N		
c_current_demo_sk	identifier			cd_demo_sk
c_current_hdemo_sk	identifier			hd_demo_sk
c_current_addr_sk	identifier			ca_addres_sk
c_first_shipto_date_sk	identifier			d_date_sk
c_first_sales_date_sk	identifier			d_date_sk
c_salutation	char(10)			
c_first_name	char(20)			
c_last_name	char(30)			
c_preferred_cust_flag	char(1)			
c_birth_day	integer			
c_birth_month	integer			
c_birth_year	integer			
c_birth_country	varchar(20)			
c_login	char(13)			
c_email_address	char(50)			
c_last_review_date_sk	identifier			d_date_sk

#### 2.4.8 Customer\_address (CA)

Each row in this table represents a unique customer address (each customer can have more than one address)

**Table 2-15 Customer\_address Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
ca_address_sk	identifier	N	Y	
ca_address_id (B)	char(16)	N		
ca_street_number	char(10)			
ca_street_name	varchar(60)			
ca_street_type	char(15)			
ca_suite_number	char(10)			
ca_city	varchar(60)			
ca_county	varchar(30)			
ca_state	char(2)			
ca_zip	char(10)			
ca_country	varchar(20)			
ca_gmt_offset	decimal(5,2)			
ca_location_type	char(20)			

#### 2.4.9 Customer\_demographics (CD)

The customer demographics table contains one row for each unique combination of customer demographic information.

**Table 2-16 Customer\_demographics Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
cd_demo_sk	identifier	N	Y	
cd_gender	char(1)			
cd_marital_status	char(1)			
cd_education_status	char(20)			
cd_purchase_estimate	integer			
cd_credit_rating	char(10)			
cd_dep_count	integer			
cd_dep_employed_count	integer			
cd_dep_college_count	integer			

#### 2.4.10 Date\_dim (D)

Each row in this table represents one calendar day. The surrogate key (d\_date\_sk) for a given row is derived from the julian date being described by the row.

**Table 2-17 Date\_dim Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
d_date_sk	identifier	N	Y	
d_date_id (B)	char(16)	N		
d_date	date			
d_month_seq	integer			
d_week_seq	integer			
d_quarter_seq	integer			
d_year	integer			
d_dow	integer			
d_moy	integer			
d_dom	integer			
d_qoy	integer			
d_fy_year	integer			
d_fy_quarter_seq	integer			
d_fy_week_seq	integer			
d_day_name	char(9)			
d_quarter_name	char(6)			
d_holiday	char(1)			
d_weekend	char(1)			

Column	Datatype	NULLs	Primary Key	Foreign Key
d_following_holiday	char(1)			
d_first_dom	integer			
d_last_dom	integer			
d_same_day_ly	integer			
d_same_day_lq	integer			
d_current_day	char(1)			
d_current_week	char(1)			
d_current_month	char(1)			
d_current_quarter	char(1)			
d_current_year	char(1)			

#### 2.4.11 Household\_demographics (HD)

Each row of this table defines a household demographic profile.

**Table 2-18 Household\_demographics Column Definition**

Column	Datatype	NULLs	Primary Key	Foreign Key
<b>hd_demo_sk</b>	identifier	N	Y	
hd_income_band_sk	identifier			ib_income_band_sk
hd_buy_potential	char(15)			
hd_dep_count	integer			
hd_vehicle_count	integer			

#### 2.4.12 Item (I)

Each row in this table represents a unique product formulation (e.g., size, color, manufacturer, etc.).

**Table 2-19 Item Column Definition**

Column	Datatype	NULLs	Primary Key	Foreign Key
<b>i_item_sk</b>	identifier	N	Y	
i_item_id (B)	char(16)	N		
i_rec_start_date	date			
i_rec_end_date	date			
i_item_desc	varchar(200)			
i_current_price	decimal(7,2)			
i_wholesale_cost	decimal(7,2)			
i_brand_id	integer			
i_brand	char(50)			
i_class_id	integer			
i_class	char(50)			
i_category_id	integer			
i_category	char(50)			
i_manufact_id	integer			
i_manufact	char(50)			
i_size	char(20)			
i_formulation	char(20)			
i_color	char(20)			
i_units	char(10)			
i_container	char(10)			
i_manager_id	integer			
i_product_name	char(50)			

#### 2.4.13 Income\_band (IB)

Each row in this table represents details of an income range.

**Table 2-20: Income\_band Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
<b>ib_income_band_sk</b>	identifier	N	Y	
ib_lower_bound	integer			
ib_upper_bound	integer			

#### 2.4.14 Promotion (P)

Each row in this table represents details of a specific product promotion (e.g., advertising, sales, PR).

**Table 2-21: Promotion Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
p_promo_sk	identifier	N	Y	
p_promo_id (B)	char(16)	N		
p_start_date_sk	identifier			d_date_sk
p_end_date_sk	identifier			d_date_sk
p_item_sk	identifier			i_item_sk
p_cost	decimal(15,2)			
p_response_target	integer			
p_promo_name	char(50)			
p_channel_dmail	char(1)			
p_channel_email	char(1)			
p_channel_catalog	char(1)			
p_channel_tv	char(1)			
p_channel_radio	char(1)			
p_channel_press	char(1)			
p_channel_event	char(1)			
p_channel_demo	char(1)			
p_channel_details	varchar(100)			
p_purpose	char(15)			
p_discount_active	char(1)			

#### 2.4.15 Reason (R)

Each row in this table represents a reason why an item was returned.

**Table 2-22: Reason Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
r_reason_sk	identifier	N	Y	
r_reason_id (B)	char(16)	N		
r_reason_desc	char(100)			

#### 2.4.16 Ship\_mode (SM)

Each row in this table represents a shipping mode.

**Table 2-23: Ship\_mode Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
sm_ship_mode_sk	identifier	N	Y	
sm_ship_mode_id (B)	char(16)	N		
sm_type	char(30)			
sm_code	char(10)			
sm_carrier	char(20)			
sm_contract	char(20)			

#### 2.4.17 Time\_dim (T)

Each row in this table represents one second.

**Table 2-24: Time\_dim Column Definitions**

Column	Datatype	NULLs	Primary Key	Foreign Key
t_time_sk	Identifier	N	Y	
t_time_id (B)	char(16)	N		
t_time	Integer			
t_hour	Integer			
t_minute	Integer			
t_second	Integer			
t_am_pm	char(2)			

Column	Datatype	NULLs	Primary Key	Foreign Key
t_shift	char(20)			
t_sub_shift	char(20)			
t_meal_time	char(20)			

#### 2.4.18 dsdgen\_version

This table is not employed during the benchmark. A flat file is generated by dsdgen (see Appendix F), and it can be helpful in assuring that the current data set was built with the correct version of the TPC-DS toolkit. It is included here for completeness.

**Table 2-25: dsdgen\_version Column Definitions**

Column	Datatype	NULLs	Foreign Key
dv_version	Varchar(16)	N	
dv_create_date	date	N	
dv_create_time	time	N	
dv_cmdline_args	Varchar(200)	N	

## 2.5 Implementation Requirements

### 2.5.1 Definition of Terms

- 2.5.1.1 The tables defined in Clause 2.3 and Clause 2.4 are referred to as base tables. The flat file data generated by DSDGen corresponding to each base table and loaded into each base table is referred to as base table data. A structure containing base table data is referred to as a base table data structure.
- 2.5.1.2 Other than the base table data structures, any database structure that contains a copy of, reference to, or data computed from base table data is defined as an auxiliary data structures (**ADS**). The data in the ADS is materialized from the base table data; references are a form of materialization. There is an essential distinction between base table data contained in a base table data structure and data contained in auxiliary data structures. Because auxiliary data structures contain *copies of, references to, or data computed from* base table data, deleting data from an auxiliary data structure does not result in the loss of base table data in that it is still contained in the base table data structure. In contrast, deleting data from a base table data structure (in the absence of copies in any auxiliary data structures) does result in the loss of base table data.
- 2.5.1.3 There are two types of auxiliary data structures: Implicit and explicit. An explicit auxiliary data structure (EADS) is created as a consequence of a directive (e.g. DDL, session options, global configuration parameters). These directives are called EADS Directives. Any ADS which is not an EADS is by definition an Implicit ADS (IADS).
- Comment:** In contrast to an implicit **ADS**, an **EADS** would not have been created without the directive.
- 2.5.1.4 The assignment of groups of rows from a table or EADS to different files, disks, or areas is defined as horizontal partitioning.
- 2.5.1.5 The assignment of groups of columns of one or more rows to files, disks, or areas different from those storing the other columns of these rows is defined as vertical partitioning.
- 2.5.1.6 A Primary Key is one or more columns that uniquely identifies a row. None of the columns that are part of the Primary Key may be nullable. A table must have no more than one Primary Key. A primary key must be enforced, e.g. by a primary key constraint.
- 2.5.1.7 A Foreign Key is a column or combination of columns used to establish and enforce a link between the data in two tables. A link is created between two tables by adding the column or columns that hold one table's Primary Key values to the other table. This column becomes a Foreign Key in the second table. A foreign key must be enforced, e.g. by a foreign key constraint. Referential Integrity is a data property whereby a Foreign Key in one table has a corresponding Primary Key in a different table.

- 2.5.1.8 The definition of primary and foreign keys is optional.
- 2.5.1.9 Whenever this specification refers to a set of primary and foreign keys it refers to the set of primary and foreign keys defined in clauses 2.3 and 2.4.

## **2.5.2 Database & Tables**

- 2.5.2.1 The database shall be implemented using a commercially available database management system (DBMS).
- 2.5.2.2 The SQL data definition statements and associated scripts used to implement the logical schema definition are defined as the **DDL**.
- 2.5.2.3 The database which is built and utilized to run the ACID tests is defined as the qualification database.
- 2.5.2.4 The database which is built and utilized for performance reporting is defined as the test database.
- 2.5.2.5 The physical clustering of records of different tables within the database is allowed as long as this clustering does not alter the logical relationships of each table.

**Comment:** The intent of this clause is to permit flexibility in the physical layout of a database and based upon the defined TPC-DS schema.

- 2.5.2.6 Table names should match those provided in Clause 2.3 and Clause 2.4. If the DBMS implementation prevents the use of the table names specified in Clause 2.3 and Clause 2.4, they may be altered provided that:
- The name changes are minimal (e.g., short prefix or suffix.)
  - The name changes have no performance impact
  - The name changes are also made to the query set, in compliance with Clause 4.2.3
- 2.5.2.7 Each table listed in Clause 2.3 and Clause 2.4, shall be implemented according to the column definitions provided above.
- 2.5.2.8 The column names used in the implementation shall match those defined for each column specified in Clause 2.3 and Clause 2.4. If the DBMS implementation prevents the use of the column names specified in Clause 2.3 and Clause 2.4, they may be altered provided:
- The name changes are the minimal changes required (e.g., short prefix or suffix or character substitution.)
  - The changed names are required to follow the documented naming convention employed in the selected DBMS
  - The names used must provide no performance benefit compared to any other names that might be chosen.
  - The identical name changes must also be made to the query set, in compliance with Clause 4.2.3
- 2.5.2.9 The columns within a given table may be implemented in any order, but all columns listed in the table definition shall be implemented and there shall be no columns added to the tables.
- 2.5.2.10 Each table column defined in Clause 2.3 and Clause 2.4 shall be implemented as discrete columns and shall be independently accessible by the data manager as a single column. Specifically:
- Columns shall not be merged. For example, C\_LOGIN and C\_EMAIL\_ADDRESS cannot be implemented as two sub-parts of a single discrete column C\_DATA.
  - Columns shall not be split. For example, P\_TYPE cannot be implemented as two discrete columns P\_TYPE\_SUBSTR1 and P\_TYPE\_SUBSTR2.
- 2.5.2.11 The database shall allow for insertion of arbitrary data values that conform to the column's datatype and optional constraints defined in accordance with Clause 2.5.13.

## **2.5.3 Explicit Auxiliary Data Structures (EADS)**

- 2.5.3.1 Except as provided in this section, replication of database objects (i.e., tables, rows or columns) is prohibited.
- 2.5.3.2 An EADS which does not include data materialized from Catalog\_Sales or Catalog\_Returns is subject to the following limitations:
- It may materialize data from no more than one **base table**.
  - It may materialize all or some of the following three items:



1. The primary key or any subset of PK columns if the PK is a compound key
2. Pointers or references to corresponding base table rows (e.g., “row IDs”).
3. At most one of the following:
  - a) A foreign key or any subset of the FK columns if the FK is a compound key
  - b) A column having a date data type
  - c) A column that is a business key

2.5.3.3 An EADS which includes data materialized from Catalog\_Sales or Catalog\_Returns may not include any data materialized from Store\_Sales, Store\_Returns, Web\_Sales, Web\_Returns or Inventory.

2.5.3.4 An EADS which materializes data from both fact and dimension tables must be the result of joining on FK – PK related columns.

2.5.3.5 An EADS which materializes data from one or more dimension tables without simultaneously materializing data from Catalog\_Sales and/or Catalog\_Returns is disallowed, unless otherwise permitted by Clause 2.5.3.2. An EADS which materializes data from one or more dimension tables must materialize at least one dimension row for every fact table row, unless the foreign key value for a dimension row is null.

**Comment:** The intent is to prohibit the creation of EADS on only dimension tables, except as allowed by clause 2.5.3.3.

2.5.3.6 The implementation of EADS may involve replication of selected data from the base tables provided that:

- All replicated data are managed by the DBMS
- All replications are transparent to all data manipulation operations

2.5.3.7 The creation of all EADS must be included in the database load test (see Clause 7.4.3). EADS may not be created or deleted during the performance test.

2.5.3.8 Partitioning

2.5.3.8.1 A logical table space is a named collection of physical storage devices referenced as a single, logically contiguous, non-divisible entity.

2.5.3.8.2 **The DDL may** include syntax that directs a table in its entirety to be stored in a particular logical table space.

2.5.3.8.3 Horizontal partitioning of **base tables** or **EADS** is allowed. If the partitioning is a function of data in the table or auxiliary data structure, the assignment shall be based on the values in the partitioning column(s). Only primary keys, foreign keys, date columns and date surrogate keys may be used as partitioning columns. If partitioning DDL uses directives that specify explicit partition values for the partitioning columns, they shall satisfy the following conditions:

- They may not rely on any knowledge of the data stored in the partitioning column(s) except the minimum and maximum values for those columns, and the definition of data types for those columns provided in Clause 2.
- Within the limitations of integer division, they shall define each partition to accept an equal portion of the range between the minimum and maximum values of the partitioning column(s).
- For date-based partitions, it is permissible to partition into equally sized domains based upon an integer granularity of days, weeks, months, or years; all using the Gregorian calendar (e.g., 30 days, 4 weeks, 1 month, 1 year, etc.). For date-based partition granularities other than days, a partition boundary may extend beyond the minimum or maximum boundaries as established in that table’s data characteristics as defined in Clause 3.4
- The directives shall allow the insertion of values of the partitioning column(s) outside the range covered by the minimum and maximum values, as required by Clause 1.5.

If any directives or DDL are used to horizontally partition data, the directives, DDL, and other details necessary to replicate the partitioning behavior shall be disclosed.

Multi-level partitioning of base tables or auxiliary data structures is allowed only if each level of partitioning satisfies the conditions stated above.

2.5.3.8.4 Vertical partitioning of **base tables** or **EADS** is allowed when meeting all of the following requirements:

- SQL DDL that explicitly partitions data vertically is prohibited.
- SQL DDL must not contain partitioning directives which influence the physical placement of data on durable media.
- The row must be logically presented as an atomic set of columns.

**Comment:** This implies that vertical partitioning which does not rely upon explicit partitioning directives is allowed. Explicit partitioning directives are those that assign groups of columns of one row to files, disks or areas different from those storing the other columns in that row.

## 2.5.4 Constraints

2.5.4.1 The use of constraints is permitted. However, if constraints are used, they shall satisfy the following requirements:

- They shall be specified using SQL. There is no specific implementation requirement. For example, CREATE TABLE, ALTER TABLE, and CREATE TRIGGER are all valid statements
- Constraints shall be enforced either at the statement level or at the transaction level
- All defined constraints shall be enforced and validated before the load test is complete (see Clause 0)
- They are limited to **primary key**, **foreign key**, and NOT NULL constraints

2.5.4.2 If foreign key constraints are defined, there is no specific requirement for a particular delete/update action when enforcing a constraint (e.g., ANSI SQL RESTRICT, CASCADE, NO ACTION, are all acceptable).

## 2.6 Data Access Transparency Requirements

2.6.1 Data Access Transparency is the property of the system that removes from the query text any knowledge of the physical location and access mechanisms of partitioned data. No finite series of tests can prove that the system supports complete data access transparency. The requirements below describe the minimum capabilities needed to establish that the system provides transparent data access. An implementation that uses horizontal partitioning shall meet the requirements for transparent data access described in Clauses 2.6.2 and 2.6.3.

**Comment:** The intent of this Clause is to require that access to physically and/or logically partitioned data be provided directly and transparently by services implemented by commercially available layers such as the interactive SQL interface, the database management system (DBMS), the operating system (OS), the hardware, or any combination of these.

2.6.2 Each of the tables described in Clause 2.3 and Clause 2.4 shall be identifiable by names that have no relationship to the partitioning of tables. All data manipulation operations in the executable query text (see Clause 3) shall use only these names.

2.6.3 Using the names which satisfy Clause 2.6.2, any arbitrary non-TPC-DS query shall be able to reference any set of rows or columns that is:

- Identifiable by any arbitrary condition supported by the underlying DBMS
- Using the names described in Clause 2.6.2 and using the same data manipulation semantics and syntax for all tables

For example, the semantics and syntax used to query an arbitrary set of rows in any one table shall also be usable when querying another arbitrary set of rows in any other table.

**Comment:** The intent of this clause is that each TPC-DS query uses general purpose mechanisms to access data in the database.

### 3 Scaling and Database Population

This clause defines the database population and how it scales.

#### 3.1 Scaling Model

3.1.1 The TPC-DS benchmark defines a set of discrete scaling points (“**scale factors**”) based on the approximate size of the raw data produced by **dsdgen**. The actual byte count may vary depending on individual hardware and software platforms.

3.1.2 The set of scale factors defined for TPC-DS is:

- 100GB, 300GB, 1TB, 3TB, 10TB, 30TB, 100TB

where gigabyte (GB) is defined to be  $2^{30}$  bytes and terabyte (TB) is defined to be  $2^{40}$  bytes.

**Comment:** The maximum size of the test database for a valid performance test is currently set at 100TB. The TPC recognizes that additional benchmark development work is necessary to allow TPC-DS to scale beyond that limit.

3.1.3 Each defined scale factor has an associated value for **SF**, a unit-less quantity, roughly equivalent to the number of gigabytes of data present in the data warehouse. The relationship between scale factors and SF is summarized in Table 3-1 Scale Factor and SF.

**Table 3-1 Scale Factor and SF**

Scale Factor	SF
100GB	100
300GB	300
1TB	1000
3TB	3000
10TB	10000
30TB	30000
100TB	100000

3.1.4 Test sponsors may choose any scale factor from the defined series. No other scale factors may be used for a TPC-DS result.

3.1.5 Results at the different scale factors are not comparable, due to the substantially different computational challenges found at different data volumes.

#### 3.2 Test Database Scaling

3.2.1 **Test database** is the database used to execute the database load test and the performance test (see Clause 7.4)

3.2.2 The required row count for each permissible scale factor and each table in the test database is detailed in Table 3-2 Database Row Counts ( $M = 10^6$ ,  $B = 10^9$ ).

**Comment:** The 1GB entries are used solely for the qualification database (see Clause 3.3.1) and are included here for ease of reference.

3.2.3 The row size information provided is an estimate, and may vary from one benchmark submission to another depending on the precise data base implementation that is selected. It is provided solely to assist benchmark sponsors in the sizing of benchmark configurations.

**Table 3-2 Database Row Counts**

Table	Avr Row	Sample Row Counts. Number of rows are within 1/100 <sup>th</sup> Percent of these numbers
-------	---------	---

	Size in bytes	1GB	100GB	300GB	1TB	3TB	10TB	30TB	100TB
call_center	305	6	30	36	42	48	54	60	60
catalog_page	139	11,718	20,400	26,000	30,000	36,000	40,000	46,000	50,000
catalog_returns	166	144,067	14,404,374	43,193,472	143,996,756	432,018,033	1,440,033,112	4,319,925,093	14,400,175,879
catalog_sales	226	1,441,548	143,997,065	431,969,836	1,439,980,416	4,320,078,880	14,399,964,710	43,200,404,822	143,999,334,399
customer	132	100,000	2,000,000	5,000,000	12,000,000	30,000,000	65,000,000	80,000,000	100,000,000
customer_address	110	50,000	1,000,000	2,500,000	6,000,000	15,000,000	32,500,000	40,000,000	50,000,000
customer_demographics	42	1,920,800	1,920,800	1,920,800	1,920,800	1,920,800	1,920,800	1,920,800	1,920,800
date_dim	141	73,049	73,049	73,049	73,049	73,049	73,049	73,049	73,049
household_demographics	21	7,200	7,200	7,200	7,200	7,200	7,200	7,200	7,200
income_band	16	20	20	20	20	20	20	20	20
inventory	16	11,745,000	399,330,000	585,684,000	783,000,000	1,033,560,000	1,311,525,000	1,627,857,000	1,965,337,830
item	281	18,000	204,000	264,000	300,000	360,000	402,000	462,000	502,000
promotions	124	300	1,000	1,300	1,500	1,800	2,000	2,300	2,500
reason	38	35	55	60	65	67	70	72	75
ship_mode	56	20	20	20	20	20	20	20	20
store	263	12	402	804	1,002	1,350	1,500	1,704	1,902
store_returns	134	287,514	28,795,080	86,393,244	287,999,764	863,989,652	2,879,970,104	8,639,952,111	28,800,018,820
store_sales	164	2,880,404	287,997,024	864,001,869	2,879,987,999	8,639,936,081	28,799,983,563	86,399,341,874	287,997,818,084
time_dim	59	86,400	86,400	86,400	86,400	86,400	86,400	86,400	86,400
warehouse	117	5	15	17	20	22	25	27	30
web_page	96	60	2,040	2,604	3,000	3,600	4,002	4,602	5,004
web_returns	162	71,763	7,197,670	21,599,377	71,997,522	216,003,761	720,020,485	2,160,007,345	7,199,904,459
web_sales	226	719,384	72,001,237	216,009,853	720,000,376	2,159,968,881	7,199,963,324	21,600,036,511	71,999,670,164
web_site	292	30	24	42	54	66	78	84	96

### 3.3 Qualification Database Scaling

3.3.1 The **Qualification database** is the database used to execute the query validation test (see Clause 7.3)

3.3.2 The intent is that the functionality exercised by running the validation queries against the qualification database be the same as that exercised against the test database during the performance test. To this end, the qualification database must be identical to the test database in virtually every regard (except size), including but not limited to:

- Column definitions
- Method of data generation and loading (but not degree of parallelism)
- Statistics gathering method
- ACID property implementation
- Type of partitioning (but not degree of partitioning)
- Replication
- Table type (if there is a choice)

h) EADS (e.g., indices)

- 3.3.3 The qualification database may differ from the test database only if the difference is directly related to the difference in sizes. For example, if the test database employs horizontal partitioning (see Clause 2.5.3.7), then the qualification database must also employ horizontal partitioning, though the number of partitions may differ in each case. As another example, the qualification database could be configured such that it uses a representative sub-set of the CPUs, memory and disks used by the test database configuration. If the qualification database configuration differs from the test database configuration in any way, the differences must be disclosed
- 3.3.4 The qualification database must be populated using **dsdgen**, and use a scale factor of 1GB.
- 3.3.5 The row counts of the qualification database are defined in Clause 3.2.

#### 3.4 dsdgen and Database Population

- 3.4.1 The test database and the qualification database must be populated with data produced by **dsdgen**, the TPC-supplied data generator for TPC-DS. The major and minor version number of **dsdgen** must match that of the TPC-DS specification. The source code for **dsdgen** is provided as part of the electronically downloadable portion of this specification (see Appendix F).
- 3.4.2 The data generated by **dsdgen** are meant to be compliant with Table 3-2 and Table 5-2. In case of differences between the table and the data generated by **dsdgen**, Table 3-2 and Table 5-2 prevail.
- 3.4.3 Vendors are allowed to modify the **dsdgen** code for both the initial database population and the data maintenance. However, the resultant data must meet the following requirements in order to be considered correct:
- a) The content of individual columns must be identical to that produced by **dsdgen**.
  - b) The data format of individual columns must be identical to that produced by **dsdgen**.
  - c) The number of rows generated for a given scale factor must be identical to that specified in Table 3-2 and Table 5-2.

If a modified version of dsdgen is used, the modified source code must be disclosed in full. In addition, the auditor must verify that the modified source code which is disclosed matches the data generation program used in the benchmark execution.

**Comment:** The intent of this clause is to allow for modification of the dsdgen code required for portability or speed, while precluding any change that affects the resulting data. Minor changes for portability or bugs are permitted in dsdgen for both initial database population and data maintenance.

- 3.4.4 If modifications are restricted to a subset of the source code, the vendor may publish only the individual dsdgen source code files which have been modified.

#### 3.5 Data Validation

The test database must be verified for correct data content. This must be done after the initial database load and prior to any performance tests. A validation data set is produced using **dsdgen** with the “-validate” and “-vcount” options. The minimum value for “-vcount” is 50, which produces 50 rows of validation data for most tables. The exceptions being the “returns” fact tables which will only have 5 rows each on average and the dimension tables with fewer than 50 total rows.

All rows produced in the validation data set must exist in the test database.

## 4 Query Overview

### 4.1 General Requirements and Definitions for Queries

#### 4.1.1 Query Definition and Availability

##### 4.1.1.1 Each query is described by the following components:

- a) A business question, which illustrates the business context in which the query could be used. The business questions are listed in Appendix B.
- b) The functional query definition, as specified in the TPC-supplied query template (see Clause 4.1.2 for a discussion of Functional Query Definitions)
- c) The substitution parameters, which describe the substitution values needed to generate the executable query text
- d) The answer set, which is used in query validation (see Clause 7.3)

**Comment:** Some functional query definitions include a limit on the number of rows to be returned by the query. These limits are omitted from the business question.

**Comment:** In cases where the business question does not accurately describe the functional query definition, the latter will prevail.

4.1.1.2 Due to the large size of the TPC-DS query set, this document does not contain all of the query components. Refer to Table 0-1 Electronically Available Specification Material for information on obtaining the query set.

#### 4.1.2 Functional Query Definitions

4.1.2.1 The functionality of each query is defined by its query template and dsqgen.

4.1.3 **dsqgen** translates the query templates into fully functional SQL, which is known as executable query text (EQT). The major and minor version number of **dsqgen** must match that of the TPC-DS specification. The source code for **dsqgen** is provided as part of the electronically downloadable portion of this specification (see Table 0-1 Electronically Available Specification Material).

4.1.3.1 The query templates are primarily phrased in compliance with SQL1999 core (with OLAP amendments). A template includes the following, non-standard additions:

- They are annotated, where necessary, to specify the number of rows to be returned
- They include substitution tags that, in conjunction with **dsqgen**, allow a single template to generate a large number of syntactically distinct queries, which are functionally equivalent

4.1.3.2 The executable query text for each query in a compliant implementation must be taken from either the functional query definition or an approved query variant (see Clause Appendix C). Except as specifically allowed in Clauses 4.2.3, **Error! Reference source not found.** and 4.2.5, executable query text must be used in full, exactly as provided by the TPC.

4.1.3.3 Any query template whose EQT does not match the functionality of the corresponding EQT produced by the TPC-supplied template is invalid.

4.1.3.4 All query templates and their substitution parameters shall be disclosed.

4.1.3.5 Benchmark sponsors are allowed to alter the precise phrasing of a query template to allow for minor differences in product functionality or query dialect as defined in Clause 4.2.3.

4.1.3.6 If the alterations allowed by Clause 4.2.3 are not sufficient to permit a benchmark sponsor to produce EQT that can be executed by the DBMS selected for their benchmark submission, they may submit an alternate query template for approval by the TPC (see Clause 4.2.3.4).

4.1.3.7 If the query template used in a benchmark submission is not identical to a template supplied by the TPC, it must satisfy the compliance requirements of Clauses 4.2.3, **Error! Reference source not found.** and 4.2.5.

## 4.2 Query Modification Methods

4.2.1 The queries must be expressed in a commercially available implementation of the SQL language. Since the ISO SQL language is continually evolving, the TPC-DS benchmark specification permits certain deviations from the SQL phrasing used in the TPC-supplied query templates.

4.2.2 There are four types of permissible deviations:

- a) Minor query modifications, defined in Clause 4.2.3
- b) Modifications to limit row counts, defined in clause **Error! Reference source not found.**
- c) Modifications for extraction queries, defined in clause 4.2.5
- d) Approved query variants, defined in Appendix C

### 4.2.3 Minor Query Modifications

4.2.3.1 It is recognized that implementations require specific adjustments for their operating environment and the syntactic variations of its dialect of the SQL language. The query modifications described in Clause 4.2.3.4:

- Are defined to be minor
- Do not require approval
- May be used in conjunction with any other minor query modifications
- May be used to modify either a functional query definition or an approved variant of that definition

Modifications that do not fall within the bounds described in Clause 4.2.3.4 are not minor and are not compliant unless they are an integral part of an approved query variant (see Appendix C).

**Comment:** The only exception is for the queries that require a given number of rows to be returned. The requirements governing this exception are given in Clause 4.2.4.1

4.2.3.2 The application of minor query modifications to functional query definitions or approved variants must be consistent over the query set. For example, if a particular vendor-specific date expression or table name syntax is used in one query, it must be used in all other queries involving date expressions or table names.

4.2.3.3 The use of minor modifications shall be disclosed and justified (see Clause 10.3.4.5).

4.2.3.4 The following query modifications are minor:

#### a) Tables:

1. Table names - The table and view names found in the CREATE TABLE, CREATE VIEW, DROP VIEW and FROM clause of each query may be modified to reflect the customary naming conventions of the system under test.
2. Tablespace references - CREATE TABLE statements may be augmented with a tablespace reference conforming to the requirements of Clause 3.
3. WITH() clause - Queries using the "with()" syntax, also known as common table sub-expressions, can be replaced with semantically equivalent derived tables or views.

#### b) Joins:

1. Outer Join - For outer join queries, vendor specific syntax may be used instead of the specified syntax. For example, the join expression "CUSTOMER LEFT OUTER JOIN ORDERS ON C\_CUSTKEY = O\_CUSTKEY" may be replaced by adding CUSTOMER and ORDERS to the from clause and adding a specially-marked join predicate (e.g., C\_CUSTKEY \*= O\_CUSTKEY).
2. Inner Join - For inner join queries, vendor specific syntax may be used instead of the specified syntax. For example, the join expression "FROM CUSTOMER, ORDERS WHERE C\_CUSTKEY = O\_CUSTKEY" may be modified to use a JOIN clause such as "FROM CUSTOMER JOIN ORDERS ON C\_CUSTKEY = O\_CUSTKEY".

#### c) Operators:



1. Explicit ASC - ASC may be explicitly appended to columns in an ORDER BY clause.
2. Relational operators - Relational operators used in queries such as "<", ">", "<>", "<=", and "=", may be replaced by equivalent vendor-specific operators, for example ".LT.", ".GT.", "!=" or "^=", ".LE.", and "==", respectively.
3. String concatenation operator - For queries which use string concatenation operators, vendor specific syntax can be used (e.g. || can be substituted with +).
4. Rollup operator - an operator of the form "rollup (x,y)" may be substituted with the following operator: "x,y with rollup". x,y are expressions.

d) Control statements:

1. Command delimiters - Additional syntax may be inserted at the end of the executable query text for the purpose of signaling the end of the query and requesting its execution. Examples of such command delimiters are a semicolon or the word "GO".
2. Transaction control statements - A CREATE/DROP TABLE or CREATE/DROP VIEW statement may be followed by a COMMIT WORK statement or an equivalent vendor-specific transaction control statement.
3. Dependent views - If an implementation is using variants involving views and the implementation only supports "DROP RESTRICT" semantics (i.e., all dependent objects must be dropped first), then additional DROP statements for the dependent views may be added.

e) Alias:

1. Select-list expression aliases - For queries that include the definition of an alias for a SELECT-list item (e.g., "AS" clause), vendor-specific syntax may be used instead of the specified syntax. Examples of acceptable implementations include "TITLE <string>", or "WITH HEADING <string>". Use of a select-list expression alias is optional.
2. GROUP BY and ORDER BY - For queries that utilize a view, nested table-expression, or select-list alias solely for the purposes of grouping or ordering on an expression, vendors may replace the view, nested table-expression or select-list alias with a vendor-specific SQL extension to the GROUP BY or ORDER BY clause. Examples of acceptable implementations include "GROUP BY <ordinal>", "GROUP BY <expression>", "ORDER BY <ordinal>", and "ORDER BY <expression>".
3. Correlation names - Table-name aliases may be added to the executable query text. The keyword "AS" before the table-name alias may be omitted.
4. Nested table-expression aliasing - For queries involving nested table-expressions, the nested keyword "AS" before the table alias may be omitted.
5. Column alias - column name alias may be added for columns in any SELECT list of an executable query text. These column aliases may be used to refer to the column in later portions of the query, such as GROUP BY or ORDER BY clauses.

f) Expressions and functions:

1. Date expressions - For queries that include an expression involving manipulation of dates (e.g., adding/subtracting days/months/years, or extracting years from dates), vendor-specific syntax may be used instead of the specified syntax. Examples of acceptable implementations include "YEAR(<column>)" to extract the year from a date column or "DATE(<date>) + 3 MONTHS" to add 3 months to a date.
2. Output formatting functions - Scalar functions whose sole purpose is to affect output formatting (such as treatment of null strings) or intermediate arithmetic result precision (such as COALESCE or CAST) may be applied to items in the outermost SELECT list of the query.
3. Aggregate functions - At large scale factors, the aggregates may exceed the range of the values supported by an integer. The aggregate functions AVG and COUNT may be replaced with equivalent vendor-specific functions to handle the expanded range of values (e.g., AVG\_BIG and COUNT\_BIG).
4. Substring Scalar Functions - For queries which use the SUBSTRING() scalar function, vendor-specific syntax may be used instead of the specified syntax. For example, "SUBSTRING(S\_ZIP, 1, 5)".

5. Standard Deviation Function - For queries which use the standard deviation function (stddev\_samp), vendor specific syntax may be used (e.g. stdev, stddev).
6. Explicit Casting - Scalar functions (such as CAST) whose sole purpose is to affect result precision for operations involving integer columns or values may be applied. The resulting syntax must have equivalent semantic behavior.
7. Mathematical functions - Vendors specific mathematical expressions may be used to implement mathematical functions in the executable query text. The replacement syntax must implement the full semantic behavior (e.g. handling for NULLs) of the mathematical functions as defined in the ISO SQL standard. For example, avg() may be replaced by average() or by a mathematical expressions such as sum()/count().
8. Date casting - Explicit casting of columns that are of the date datatype, as defined in Clause 2.2.2, and date constant strings, expressed in month, day and year, into a datatype that allows for date arithmetic in expressions is permissible. Replacement syntax must have equivalent semantic behavior.
9. Casting syntax: - Vendor specific casting syntax may be used to implement casting functions present in the executable query text provided that the vendor specific casting syntax is semantically equivalent to the syntax provided in the executable query text.

g) General

1. Delimited identifiers - In cases where identifier names conflict with reserved words in a given implementation, delimited identifiers may be used.
2. Parentheses - Adding or removing parentheses around expressions and sub-queries is allowed. Both an opening parenthesis '(' and its corresponding closing parenthesis ')' must be added or removed together.

**Comment:** The application of minor query modifications must result in queries that have equivalent ISO SQL semantic behavior as the queries generated from the TPC-supplied query templates.

**Comment:** These query modifications are labeled minor based on the assumption that they do not significantly impact the performance of the queries

#### 4.2.4 Row Limit Modifications

- 4.2.4.1 Some queries require that a given number of rows be returned (e.g., “Return the first 10 selected rows”). If N is the number of rows to be returned, the query must return exactly the first N rows unless fewer than N rows qualify, in which case all rows must be returned. There are four permissible ways of satisfying this requirement:
- Vendor-specific control statements supported by a test sponsor’s interactive SQL interface may be used (e.g., SET ROWCOUNT n) to limit the number of rows returned.
  - Control statements recognized by the implementation specific layer (see Clause 8.2.4) and used to control a loop which fetches the rows may be used to limit the number of rows returned (e.g., while rowcount <= n).
  - Vendor-specific SQL syntax may be added to the SELECT statement of a query template to limit the number of rows returned (e.g., SELECT FIRST n). This syntax is not classified as a minor query modification since it completes the functional requirements of the functional query definition and there is no standardized syntax defined. In all other respects, the query must satisfy the requirements of Clause 4.1.2. The syntax added must deal solely with the size of the answer set, and must not make any additional explicit reference, for example, to tables, indices, or access paths.
  - Enclosing the outer most SQL statement (or statements in case of iterative OLAP queries) with a select clause and a row limiting predicate. For example, if Q is the original query text. Then the modification would be: SELECT \* FROM (Q) where rownum<=n. This syntax is not classified as a minor query modification since it completes the functional requirements of the functional query definition and there is no standardized syntax defined. In all other respects, the query must satisfy the requirements of Clause 4.1.2. The syntax added must deal solely with the size of the answer set, and must not make any additional explicit reference, for example, to tables, indices, or access paths.

A test sponsor must select one of these methods and use it consistently for all the queries that require that a specified number of rows be returned.

#### 4.2.5 Extract Query Modifications

4.2.5.1 Some queries return large result sets. These queries correspond to the queries described in Clause 1.4 as those that produce large result sets for extraction; the results are to be saved for later analysis. The benchmark allows for alternative methods for a DBMS to extract these result rows to files in addition to the normal method of processing them through a SQL front-end tool and using the front-end tool to output the rows to a file. If a query for any stream returns 10,000 or more result rows, the vendor may extract the rows for that query in all streams to files using one of the following permitted vendor-specific extraction tools or methods:

- Vendor-specific SQL syntax may be added to the SELECT statement of a query template to redirect the rows returned to a file. For example, “Unload to file ‘outputfile’ Select c1, c2 ...”
- Vendor-specific control statements supported by a test sponsor’s interactive SQL interface may be used. For example,

```
set output_file = 'outputfile'

select c1, c2...;

unset output_file;
```

- Control statements recognized by the implementation specific layer (see Clause 8.2.4) and used to invoke an extraction tool or method.

4.2.5.2 If one of these alternative extract options is used, the output shall be formatted as delimited or fixed-width ASCII text.

4.2.5.3 If one of these alternative extract options is used, they must meet the following conditions:

A test sponsor may select only one of the options in 4.2.5.1. That method must be used consistently for all the queries that are eligible as extract queries.

- If the extraction syntax modifies the query SQL, in all other respects the query must satisfy the requirements of Clause 4.1.2. The syntax added must deal solely with the extraction tool or method, and must not make any additional explicit reference, for example, to tables, indices, or access paths.
- The test sponsor must demonstrate that the file names used, and the extract facility itself, does not provide hints or optimizations in the DBMS such that the query has additional performance gains beyond any benefits from accelerating the extraction of rows.

The tool or method used must meet all ACID requirements for the queries used in combination with the tool or method.

#### 4.2.6 Query Variants

4.2.6.1 A **Query Variant** is an alternate query template, which has been created to allow a vendor to overcome specific functional barriers or product deficiencies that could not be address by minor query modifications.

4.2.6.2 Approval of any new query variant is required prior to using such variant to produce compliant TPC-DS results. The approval process is defined Clause 4.2.7.

4.2.6.3 Query variants that have already been approved are summarized in Appendix C.

**Comment:** Since the soft appendix is updated each time a new variant is approved, test sponsors should obtain the latest version of this appendix prior to implementing the benchmark. See Appendix F **Tool Set Requirements** for more information)

#### 4.2.7 Query Variant Approval

- 4.2.7.1 New query variants will be considered for approval if they meet one of the following criteria:
- a) The vendor requesting the variant cannot successfully run the executable query text against the qualification database using the functional query definition or an approved variant even after applying appropriate minor query modifications as per Clause 4.2.3.
  - b) The proposed variant contains new or enhanced SQL syntax, relevant to the benchmark, which is defined in an Approved Committee Draft of a new ISO SQL standard.
  - c) The variant contains syntax that brings the proposed variant closer to adherence to an ISO SQL standard.
  - d) The proposed variant contains minor syntax differences that have a straightforward mapping to ISO SQL syntax used in the functional query definition and offers functionality substantially similar to the ISO SQL standard.
- 4.2.7.2 To be approved, a proposed variant should have the following properties. Not all of the properties are specifically required. Rather, the cumulative weight of each property satisfied by the proposed variant will be the determining factor in approving the variant.
- a) Variant is syntactic only, seeking functional compatibility and not performance gain.
  - b) Variant is minimal and restricted to correcting a missing functionality.
  - c) Variant is based on knowledge of the business question rather than on knowledge of the system under test (SUT) or knowledge of specific data values in the test database.
  - d) Variant has broad applicability among different vendors.
  - e) Variant is non procedural.
  - f) Variant is an approved ISO SQL syntax to implement the functional query definition.
  - g) Variant is sponsored by a vendor who can implement it and who intends on using it in an upcoming implementation of the benchmark.
- 4.2.7.3 To be approved, the proposed variant shall conform to the implementation guidelines defined in Clause 4.2.8 and the coding standards defined in Clause 4.2.9.
- 4.2.7.4 Approval of proposed query variants will be at the sole discretion of the TPC-DS subcommittee, subject to TPC policy.
- 4.2.7.5 All proposed query variants that are submitted for approval will be recorded, along with a rationale describing why they were or were not approved.
- 4.2.8 Variant Implementation Guidelines
- 4.2.8.1 When a proposed query variant includes the creation of a table, the datatypes shall conform to Clause 2.2.2.
- 4.2.8.2 When a proposed query variant includes the creation of a new entity (e.g., cursor, view, or table) the entity name shall ensure that newly created entities do not interfere with other query sessions and are not shared between multiple query sessions.
- 4.2.8.3 Any entity created within a proposed query variant must also be deleted within that variant.
- 4.2.8.4 If CREATE TABLE statements are used within a proposed query variant, they may include a tablespace reference (e.g., IN <tablespacename>). A single tablespace must be used for all tables created within a proposed query variant.
- 4.2.9 Coding Style
- 4.2.9.1 Implementers may code the executable query text in any desired coding style, including
- a) use of line breaks, tabs or white space
  - b) choice of upper or lower case text
- 4.2.9.2 The coding style used shall have no impact on the performance of the system under test, and must be consistently applied throughout the entire query set.
- Comment:** The auditor may require proof that the coding style does not affect performance.

### 4.3 Substitution Parameter Generation

- 4.3.1 Each query has one or more substitution parameters. Dsqgen must be used to generate executable query texts for the query streams. In order to generate the required number of query streams, dsqgen must be used with the RNGSEED, INPUT and STREAMS options. The value for the RNGSEED option, <SEED>, is selected as the timestamp of the end of the database load time (Load End Time) expressed in the format mmddhhmmss as defined in Clause 7.4.3.8. The value for the STREAMS option, <S>, is two times the number of streams,  $S_q$ , to be executed during each Query Run ( $S=2 * S_q$ ). The value of the INPUT option, <input.txt>, is a file containing the location of all 99 query templates in numerical order.

**Comment:** RNGSEED guarantees that the query substitution parameter values are not known prior to running the power and throughput tests. Called with a value of <S> for the STREAMS parameter, dsqgen generates  $S+1$  files, named query\_0.sql through query\_[S].sql. Each file contains a different permutation of the 99 queries.

- 4.3.2 Query\_0.sql is the sequence of queries to be executed during the Power Test; files query\_1.sql through query\_[ $S_q$ ].sql are the sequences of queries to be executed during the first Query Run; and files query\_[ $S_q+1$ ].sql through query\_[ $2*S_q$ ].sql are the sequences of queries to be executed during the second Query Run.

**Comment:** The substitution parameter values for the qualification queries are provided in 11.9.1.1 Appendix B:. They must be manually inserted into the query templates.

## 5 Data Maintenance

### 5.1 Implementation Requirements and Definitions

- 5.1.1 Data maintenance operations are performed as part of the benchmark execution. These operations consist of processing refresh data sets. The total number of refresh data sets in the benchmark equals the number of query streams in one Query Run. All data maintenance functions defined in Clause 5.3 are executed in each refresh data set. Each refresh set has its own data set as generated by dsdgen and must be used in the order generated by dsdgen. Data maintenance operations may execute concurrently with queries or alone. Refresh sets do not overlap; at most one refresh set is running at any time.
- 5.1.2 Each refresh set includes all data maintenance functions defined in Clause 5.3 on the refresh data defined in Clause 5.2. All data maintenance functions need to have finished on refresh data set  $n$  before any data maintenance function can commence on refresh data set  $n+1$  (See Clause 7.4.7.8.)
- 5.1.3 Processing of a given refresh set can only start once a certain number of queries have completed in the throughput run. The first refresh set can only start after  $(3 \cdot S_q)$  queries have completed (across all streams). Each subsequent refresh set can start after completion of an additional 192 queries (See Clause 7.4.9.5 and 7.4.9.6).

**Comment:** The purpose of linking data maintenance operations to completion of queries is so that the updates are interspersed among execution of queries in the throughput runs, although concurrent execution of updates and queries is not required.

- 5.1.4 Data maintenance functions can be decomposed or combined into any number of database transactions and the execution order of the data maintenance functions can be freely chosen as long as the following conditions are met. Particularly, the DM functions in each refresh set may be run sequentially or in parallel.
- a) ACID properties (See Clause 6.1 );
  - b) All primary/foreign key relationships must be preserved across transaction boundaries regardless of whether they have been enforced by constraint (see Clause 0). This does not imply that referential integrity constraints must be defined explicitly (Clause 0).
  - c) A time-stamped output message is sent when the data maintenance process is finished.

**Comment:** The intent of this clause is to maintain primary and foreign key referential integrity.

- 5.1.5 All existing and enabled EADS affected by any data maintenance operation must be updated atomically within those data maintenance operations. All committed updates performed by the refresh process must be visible to queries that start after the updates are committed.
- 5.1.6 The data maintenance functions must be implemented in SQL or procedural SQL. The proper implementation of the data maintenance function must be validated by the auditor who may request additional tests to ascertain that the data maintenance functions were implemented and executed in accordance with the benchmark requirements.
- 5.1.7 The key phrases “Begin Transaction” and “End Transaction”, defined in Clause 5.3, encompass the largest transaction boundaries.
- 5.1.8 The **staging area** is an optional collection of database objects (e.g. tables, indexes, views, etc.) used to implement the data maintenance functions. Database objects created in the staging area can only be used during execution of the data maintenance phase and cannot be used during any other phase of the benchmark. Any object created in the staging area needs to be disclosed in the FDR.
- 5.1.9 Any disk storage used for the staging area must be priced. Any mapping or virtualization of disk storage must be disclosed.

## 5.2 Refresh Data

- 5.2.1 The refresh data consists of a series of refresh data sets, numbered 1, 2, 3...n. <n> is identical to the number of streams used in the Query Runs of the benchmark. Each refresh data set consists of <N> flat files. The content of the flat files can be used to populate the source schema, defined in Appendix A. However, populating the source schema is not mandated. The flat files generated for each refresh data set and their corresponding source schema tables are denoted in the following table.

**Table 5-1 Flat File to Source Schema Table Mapping and Flat File Size at Scale Factor 1**

Flat File Name	Approximate Size at SF=1 <sup>1</sup>		Source Schema Table Name
	Bytes	Number of rows	
s_call_center.dat	44	1	s_call_center
s_catalog_order.dat	116505	682	s_catalog_order
s_catalog_order_lineitem.dat	592735	6138	s_catalog_order_lineitem
s_catalog_page.dat	21155	150	s_catalog_page
s_catalog_returns.dat	112182	578	s_catalog_returns
s_customer.dat	1566843	5000	s_customer
s_customer_address.dat	2349	25	s_customer_address
s_inventory.dat	26764259	540000	s_inventory
s_item.dat	88420	500	s_item
s_promotion.dat	646	5	s_promotion
s_purchase.dat	142552	1022	s_purchase
s_purchase_lineitem.dat	1312480	12264	s_purchase_lineitem
s_store.dat	86	1	s_store
s_store_returns.dat	159306	1235	s_store_returns
s_warehouse.dat	43	1	s_warehouse
s_web_order.dat	43458	256	s_web_order
s_web_order_lineitem.dat	324160	3072	s_web_order_lineitem
s_web_page.dat	81	1	s_web_page
s_web_returns.dat	42165	295	s_web_returns
s_web_site.dat	62	1	s_web_site
s_zip_to_gmt.dat	994200	99400	s_zip_to_gmt
inventory_delete	66	3	inventory_delete
delete	66	3	delete

---

<sup>1</sup> The number of rows are correct to within 0.001%. However, the number of bytes can vary from update set to update set due to NULL values.

**Table 5-2 Approximate Number of rows in the update sets**

Source Schema Table Name Flat File Name (with .dat extension)	Approximate Number of Rows <sup>2</sup> at Scale Factors:							
	1	100	300	1000	3000	10000	30000	100000
delete_1.dat	3	3	3	3	3	3	3	3
inventory_delete_1.dat	3	3	3	3	3	3	3	3
s_call_center_1.dat	1	1	1	1	1	1	1	1
s_catalog_order_1.dat	682	68104	204318	681062	2043188	6810626	20431878	68106258
s_catalog_order_lineitem_1.dat	6138	612936	1838862	6129558	18388692	61295634	183886902	612956322
s_catalog_page_1.dat	150	240	240	240	240	240	240	240
s_catalog_returns_1.dat	595	61098	183503	612485	1838772	6128994	18382810	61291609
s_customer_1.dat	5000	20000	50000	120000	300000	650000	800000	1000000
s_customer_address_1.dat	25	100	250	600	1500	3250	4000	5000
s_inventory_1.dat	270000	9180000	13464000	18000000	23760000	30150000	37422000	90360000
s_item_1.dat	500	1000	1300	1500	1800	2000	2300	2500
s_promotion_1.dat	5	10	13	15	18	20	23	25
s_purchase_1.dat	1022	102160	306480	1021594	3064780	10215938	30647816	102159386
s_purchase_lineitem_1.dat	12264	1225920	3677760	12259128	36777360	122591256	367773792	1225912632
s_store_1.dat	1	2	4	5	6	7	8	9
s_store_returns_1.dat	1200	122279	368092	1226054	3676450	12259852	36777217	122600683
s_warehouse_1.dat	1	1	1	1	1	1	1	1
s_web_order_1.dat	256	25540	76620	255398	766196	2553984	7661954	25539846
s_web_order_lineitem_1.dat	3072	306480	919440	3064776	9194352	30647808	91943448	306478152
s_web_page_1.dat	6	200	260	300	360	400	460	500
s_web_returns_1.dat	320	30796	92380	306222	918594	3061569	9190618	30642220
s_web_site_1.dat	1	1	1	1	1	1	1	1
s_zip_to_gmt_1.dat	99400	99400	99400	99400	99400	99400	99400	99400

**Table 5-3 Approximate size of update data sets in bytes**

Source Schema Table Name Flat File Name (with .dat extension)	Approximate Number of Bytes <sup>3</sup> at Scale Factors:							
	1	100	300	1000	3000	10000	30000	100000
s_call_center	44	99	44	95	44	44	106	106
s_catalog_order	116505	11764773	35419113	118319211	356209093	1189890226	3582266543	11966927381
s_catalog_order_lineitem	592735	60682587	183190789	613833353	185302876	6200096417	18729687588	62665689954
s_catalog_page	21155	34422	34137	33450	34328	34065	33951	34211
s_catalog_returns	112182	11995241	36128085	120659364	363309171	1212531153	3648947224	12186641092
s_customer	1566843	6269955	15678473	37621039	94055447	203781981	250809222	313514345
s_customer_address	2349	9352	23388	56059	140270	304292	374586	468027
s_inventory	2676425	90995637	133460080	178422606	235517360	2988571049	3709394541	4478391128
s_item	88420	177578	230451	270009	318801	353301	405306	444630
s_promotion	646	1227	1603	1790	2253	2500	2856	3101
s_purchase	142552	14444806	43549173	145457806	438594877	1464772384	41069025492	14749907338
s_purchase_lineitem	1312480	13353769	403369986	134788326	407034160	13601008735	41069025492	137232918564
s_store	86	181	354	437	551	609	716	817
s_store_returns	159306	16324150	49450552	165441528	501145568	1677710639	5088325652	17029150799
s_warehouse	43	43	43	43	43	43	43	43
s_web_order	43458	4400592	13230508	44295571	133116152	445523894	1338776975	4480621920
s_web_order_lineitem	324160	32938398	99077152	332423806	999959825	3354924415	10091449245	33855757519
s_web_page	482	15986	20817	24016	28815	31982	36801	40013
s_web_returns	42165	4443185	13344109	44803099	134594520	450275312	1353093091	4533145920
s_web_site	62	62	62	62	62	62	62	62
s_zip_to_gmt	994200	994200	994200	994200	994200	994200	994200	994200
inventory_delete	66	66	66	66	66	66	66	66
delete	66	66	66	66	66	66	66	66

<sup>2</sup> The number of rows are correct to within 0.001%.

<sup>3</sup> The number of bytes can vary from update set to update set due to NULL values.



- 5.2.2 The number of rows present in each refresh set at scale factor 1 for each of the flat files is summarized in Table 5-1.
- 5.2.3 The refresh data set of each data maintenance function must be generated using **dsgen**. The execution of **dsgen** is not timed. The output of **dsgen** is a text file. The storage to hold the refresh data sets must be part of the priced configuration.
- 5.2.4 The refresh data set produced by dsgen can be modified in the following way: The output file for each table of the refresh data set can be split into n files where each file contains approximately 1/n of the total number of rows of the original output file. The order of the rows in the original output file must be preserved, such that the concatenation of all n files is identical to the original file.
- 5.2.5 Reading the refresh data is a timed part of the data maintenance process. The data set for a specific refresh stream must be loaded and timed as part of the execution of the refresh stream. The loading of data must be performed via generic processes inherent to the DBMS or by the loader utility the database software provides and supports for general data loading. It is explicitly prohibited to use a loader tool that has been specifically developed for TPC-DS.
- 5.3 Data Maintenance Functions
- 5.3.1 Data maintenance functions perform insert, update and delete operations that are defined in pseudo code. Depending on which operation they perform and on which type of table, they are categorized as Method1 through Method5. They are:
- Method 1: non history keeping dimension data maintenance
- Method 2: history keeping dimension data maintenance
- Method 3: fact insert data maintenance
- Method 4: fact delete data maintenance
- Method 5: inventory delete data maintenance
- 5.3.2 The following table lists all data maintenance functions, their type of operation and target table. The number of rows in the views must be equal to the rowcounts in the source schema tables listed in column 6 of Table 5-4. The rowcounts of the source schema tables are listed in Table 5-2.

**Table 5-4 Data Maintenance Function Summary**

Data Maintenance Function ID	Data Maintenance Function	Type of Operation	View Name	Target Table	Source Schema Table
1	DM_CP(Clause 5.3.13.7)	Method 1	catv	catalog_page	s_catalog_page
2	DM_C (Clause 5.3.13.4)	Method 1	custv	customer	s_customer
3	DM_CA (Clause 5.3.13.3)	Method 1	cadv	customer address	s_customer
4	DM_P(Clause 5.3.13.8)	Method 1	promv	promotion	s_promotion
5	DM_W(Clause 5.3.13.5)	Method 1	wrhsv	warehouse	s_warehouse
6	DM_CC(Clause 5.3.13.6)	Method 2	ccv	call_center	s_call_center
7	DM_I(Clause 5.3.13.2)	Method 2	itemv	item	s_item
8	DM_S (Clause 5.3.13.1)	Method 2	stov	store	s_store
9	DM_WP(Clause 5.3.13.10)	Method 2	webv	web_page	s_web_page
10	DM_WS(Clause 5.3.13.9)	Method 2	websv	web_site	s_web_site
11	LF_CR(Clause 5.3.13.16)	Method 3	crv	catalog_returns	s_catalog_returns
12	LF_CS(Clause 5.3.13.15)	Method 3	csv	catalog_sales	s_catalog_sales
13	LF_I(Clause 5.3.13.17)	Method 3	iv	inventory	s_inventory
14	LF_SR(Clause 5.3.13.12)	Method 3	srv	store_returns	s_store_returns
15	LF_SS(Clause 5.3.13.11)	Method 3	ssv	store_sales	s_purchase_lineitem

Data Maintenance Function ID	Data Maintenance Function	Type of Operation	View Name	Target Table	Source Schema Table
16	LF_WR(Clause 5.3.13.14)	Method 3	wrv	web_returns	s_web_returns
17	LF_WS(Clause 5.3.13.13)	Method 3	wsv	web_sales	s_web_order_lineitem
18	DF_CS(Clause 5.3.13.20)	Method 4	-	catalog_sales [S], catalog_returns [R]	-
19	DF_SS(Clause 5.3.13.19)	Method 4	-	store_sales [S], store_returns [R]	-
20	DF_WS(Clause 5.3.13.21)	Method 4	-	web_sales [S], web_returns [R]	-
21	DF_I(Clause 5.3.13.22)	Method 5	-	Inventory [I]	-

5.3.3 Data maintenance function methods 1, 2 and 3 read rows from a view V (see column View Name of table in Clause 5.3.2) and insert or update rows into a data warehouse table T. Both V and T are defined as part of the data maintenance function. T is created as part of the initial load of the data warehouse. V is a logical table that does not need to be instantiated.

5.3.4 The primary key of V is defined in the data maintenance function. Each data maintenance function contains a table with column mapping between its view V and its data warehouse table T. The primary key of V is denoted in bold letters on the left side of this mapping table (e.g. Table 5-5).

5.3.5 Business keys are the primary keys from the source schema. Business keys are denoted in bold letters on the right side of the mapping table for the data maintenance function (e.g. Table 5-5). Type 1 business keys are denoted with a (1) after the business key. Type 2 business keys are denoted with a (2) after the business key. Type 1 business keys are those of non-history keeping dimensions. Type 2 business keys are those of history keeping dimensions.

5.3.6 Generating a new primary key value for a dimension table is defined as generating the next largest value in the dense sequence of the table's primary key values. That is, assuming the largest current primary key value is x then the next value is x+1.

5.3.7 Method 1: non history keeping  
Dimension Type: Non-Historical

```
for every row v in view V
  begin transaction /* minimal transaction boundary */
  if there is a row d in table D where the business keys of v and d are equal
    replace all non-primary key values of d with values from v
  end-if
end transaction
end-for
```

5.3.8 Method 2: history keeping  
Dimension Type: Historical

```
for every row v in view V corresponding to dimension table D
  get row v into local variable lv
  begin transaction /* minimal transaction boundary */
  if there is a row d in D where the business keys of v and d are equal
    get the row d of D where the value of d.rec_end_date is NULL
    update rec_end_date of d with current date minus one day
    update rec_start_date of lv with current date
  end-if
  generate next primary key value pkv of D
  insert lv into D including pkv as primary key and NULL as rec_end_date
end transaction
end-for
```

5.3.9 Method 3: Fact Table Load

```

for every row v in view V corresponding to fact table F
    get row v into local variable lv
    begin transaction /* minimal transaction boundary */
    for every type 1 business key column bkc in v
        get row d from dimension table D corresponding to bkc
        where the business keys of v and d are equal
        update bkc of lv with surrogate key of d
    end for
    for every type 2 business key column bkc in v
        get row d from dimension table D corresponding to bkc
        where the business keys of v and d are equal and
        rec_end_date is NULL
        update bkc of lv with surrogate key of d
    end for
    insert lv into F
    end transaction
end for

```

#### 5.3.10 Method 4: Sales and Returns Fact Table Delete

```

Delete rows from R with corresponding rows in S
    where d_date between Date1 and Date2
Delete rows from S
    where d_date between Date1 and Date2

```

**Comment:** D\_date is a column of the date\_dim dimension. D\_date has to be obtained by joining to the date\_dim dimension on sales date surrogate key. The sales date surrogate key for the store sales is ss\_sold\_date\_sk, for catalog it is cs\_sold\_date\_sk and for web sales it is ws\_sold\_date\_sk. The minimal database transaction boundary is the combination of sales and return events as defined by ticket\_number for store\_sales and order\_number for catalog and web sales. The intent is to delete corresponding sales and returns as one unit of work.

#### 5.3.11 Method 5: Inventory Fact Table Delete

```

Delete rows from I where d_date between Date1 and Date2

```

**Comment:** D\_date is a column of the date\_dim dimension. D\_date has to be obtained by joining to the date\_dim dimension on inv\_date\_sk. The minimal database transaction boundary is one row.

#### 5.3.12 Each data maintenance function inserting or updating rows in dimension and fact tables is defined by the following components:

- Descriptor**, indicating the name of the data maintenance function in the form of DM\_<abbreviation of data warehouse table> for dimensions and LF\_<abbreviation of the data warehouse fact table> for fact tables. The extension indicates the data warehouse table that is populated with this data maintenance function.
- The **data maintenance method** describes the pseudo code of the data maintenance function.
- A **SQL view V** describing which tables of the source schema need to be joined to obtain the correct rows to be loaded.
- The **column mapping** defining which source schema columns map to which data warehouse columns;

#### 5.3.13 Each data maintenance function deleting rows from fact tables is defined by the following components:

- Descriptor**, indicating the name of the data maintenance function in the form of DF\_<abbreviation of data warehouse fact table>. The extension indicates the data warehouse fact table from which rows are deleted.
- Tables: **S** and **R**, or **I** in case of inventory
- Two dates: **Date1** and **Date2**
- The **data maintenance method** indicates how data is deleted

**Comment:** In the flat files generated by dsdgen for data maintenance there are 2 files which relate to deletes. One flat file (delete\_<n>.dat) associated with deletes applies to sales and returns for store, web and catalog where <n> denotes the set number, defined in Clause 5.1.2). The second flat file (inventory\_delete\_<n>.dat) applies to inventory only where <n> denotes the set number, defined in Clause 5.1.2). In each delete flat file there are 3 sets of start and end dates for the delete function. Each of the 3 sets of dates must be applied.

#### 5.3.13.1 DM\_S

```
CREATE view storv AS
SELECT next value for store_seq s_store_sk
,stor_store_id s_store_id
,current_date s_rec_start_date
,cast(NULL as date) s_rec_end_date
,d1.d_date_sk s_closed_date_sk
,stor_name s_store_name
,stor_employees s_number_employees
,stor_floor_space s_floor_space
,stor_hours s_hours
,stor_store_manager s_manager
,stor_market_id s_market_id
,stor_geography_class s_geography_class
,s_market_desc
,stor_market_manager s_market_manager
,s_division_id
,s_division_name
,s_company_id
,s_company_name
,s_street_number
,s_street_name
,s_street_type
,s_suite_number
,s_city
,s_county
,s_state
,s_zip
,s_country
,s_gmt_offset
,stor_tax_percentage s_tax_percentage
FROM s_store
LEFT OUTER JOIN store
ON (stor_store_id = s_store_id AND s_rec_end_date is null)
LEFT OUTER JOIN date_dim d1 ON (cast(stor_closed_date as date)= d1.d_date);
```

**Table 5-5: Column mapping for the store dimension**

Source Schema Column	Target Column
Generated as described in Method2	s_store_sk
<b>s_store_id(2)</b>	<b>s_store_id</b>
Generated as described in Method2	s_rec_start_date
Generated as described in Method1	s_rec_end_date
s_close_date_sk	s_close_date_sk
s_store_name	s_store_name
s_number_employees	s_number_employees
s_floor_space	s_floor_space
s_hours	s_hours
s_manager	s_manager
s_market_id	s_market_id
s_geography_class	s_geography_class
s_market_desc	s_market_desc
s_market_manager	s_market_manager
s_division_id	s_division_id
s_division_name	s_division_name
s_company_id	s_company_id
s_company_name	s_company_name
s_street_number	s_street_number
s_street_name	s_street_name
s_street_type	s_street_type

Source Schema Column	Target Column
s_suite_number	s_suite_number
s_city	s_city
s_county	s_county
s_state	s_state
s_zip	s_zip
s_country	s_country
s_gmt_offset	s_gmt_offset
s_tax_percentage	s_tax_percentage

### 5.3.13.2 DM\_I

```

CREATE VIEW itemv as
SELECT next value for item_seq i_item_sk
, item_item_id i_item_id
, current_date i_rec_start_date
, cast(NULL as date) i_rec_end_date
, item_item_description i_item_desc
, item_list_price i_current_price
, item_wholesale_cost i_wholesalecost
, i_brand_id
, i_brand
, i_class_id
, i_class
, i_category_id
, i_category
, i_manufact_id
, i_manufact
, item_size i_size
, item_formulation i_formulation
, item_color i_color
, item_units i_units
, item_container i_container
, item_manager_id i_manager
, i_product_name
FROM s_item
LEFT OUTER JOIN item ON (item_item_id = i_item_id and i_rec_end_date is null);

```

**Table 5-6: Column mapping for the item dimension**

Source Schema Column	Target Column
Generated as described in generic algorithm	i_item_sk
<b>i_item_id(2)</b>	<b>i_item_id</b>
Generated as described in generic algorithm	i_rec_start_date
Generated as described in generic algorithm	i_rec_end_date
i_item_desc	i_item_desc
i_current_price	i_current_price
i_brand_id	i_brand_id
i_brand	i_brand
i_class_id	i_class_id
i_class	i_class
i_category	i_category
i_category_id	i_category_id
i_manufact	i_manufact
i_manufact_id	i_manufact_id
i_size	i_size
i_formulation	i_formulation
i_color	i_color
i_units	i_units
i_container	i_container
i_manager_id	i_manager_id
i_product_name	i_product_name

### 5.3.13.3 DM\_CA

```
CREATE VIEW cadrv AS
SELECT ca_address_sk
      ,ca_address_id
      ,cust_street_number ca_street_number
      ,rtrim(cust_street_name1) || ' ' || rtrim(cust_street_name2) ca_street_name
      ,cust_street_type ca_street_type
      ,cust_suite_number ca_suite_number
      ,cust_city ca_city
      ,cust_county ca_county
      ,cust_state ca_state
      ,cust_zip ca_zip
      ,cust_country ca_country
      ,zipg_gmt_offset ca_gmt_offset
      ,cust_loc_type ca_location_type
FROM s_customer,
     customer customer,
     customer_address cat,
     s_zip_to_gmt
WHERE cust_customer_id = c_customer_id
      AND c_current_addr_sk = ca_address_sk
      AND cust_zip = zipg_zip;
```

**Table 5-7: Column mapping for the customer\_address dimension**

Source Schema Column	Target Column
Generated as described in Method1	ca_address_sk
<b>cust.cust_customer_id(1)</b>	<b>ca_address_id</b>
cust.cust_street_number	ca_street_number
concat(custv.cust_street_name1,custv.cust_street_name1)	ca_street_name
cust.cust_street_type	ca_street_type
cust.cust_suite_number	ca_suite_number
cust.cust_city	ca_city
cust.cust_county	ca_county
cust.cust_state	ca_state
cust.cust_zip	ca_zip
cust.cust_country	ca_country
cust.zipg_gmt_offset	ca_gmt_offset

### 5.3.13.4 DM\_C

```
CREATE view custv as
SELECT c_customer_sk
      ,cust_customer_id c_customer_id
      ,cd_demo_sk c_current_cdemo_sk
      ,hd_demo_sk c_current_hdemo_sk
      ,ca_address_sk c_current_addr_sk
      ,d1.d_date_sk c_first_shipto_date_sk
      ,d2.d_date_sk c_first_sales_date_sk
      ,cust_salutation c_salutation
      ,cust_first_name c_first_name
      ,cust_last_name c_last_name
      ,cust_preferred_flag c_preferred_cust_flag
      ,extract(day FROM cast(cust_birth_date as date)) c_birth_day
      ,extract(month FROM cast(cust_birth_date as date)) c_birth_month
      ,extract(year FROM cast(cust_birth_date as date)) c_birth_year
      ,cust_birth_country c_birth_country
      ,cust_login_id c_login
      ,cust_email_address c_email_address
      ,cust_last_review_date c_last_review_date
FROM s_customer
LEFT OUTER JOIN customer on (c_customer_id=cust_customer_id)
LEFT OUTER JOIN customer_address on (c_current_addr_sk = ca_address_sk)
      ,customer_demographics
      ,household_demographics
      ,income_band ib
      ,date_dim d1
      ,date_dim d2
WHERE cust_gender = cd_gender
```

```

and cust_marital_status = cd_marital_status
and cust_educ_status = cd_education_status
and cust_purch_est = cd_purchase_estimate
and cust_credit_rating = cd_credit_rating
and cust_depend_cnt = cd_dep_count
and cust_depend_emp_cnt = cd_dep_employed_count
and cust_depend_college_cnt = cd_dep_college_count
and round(cust_annual_income, 0) between ib.ib_lower_bound and ib.ib_upper_bound
and hd_income_band_sk = ib_income_band_sk
and cust_buy_potential = hd_buy_potential
and cust_depend_cnt= hd_dep_count
and cust_vehicle_cnt = hd_vehicle_count
and d1.d_date = cust_first_purchase_date
and d2.d_date = cust_first_shipto_date;

```

**Table 5-8: Column mapping for the customer dimension**

Source Schema Column	Target Column
Generated as described in generic algorithm	c_customer_sk
<b>custv.cust_customer_id(1)</b>	<b>c_customer_id</b>
custv.cd_demo_sk	c_current_cdemo_sk
custv.hd_demo_sk	c_current_hdemo_sk
custv.ca_address_sk	c_current_addr_sk
custv.fpdata	c_first_shipto_date_sk
custv.fsdate	c_first_sales_date_sk
custv.cust_salutation	c_salutation
custv.cust_first_name	c_first_name
custv.cust_last_name	c_last_name
custv.cust_preferred_flag	c_preferred_cust_flag
custv.bday	c_birth_day
custv.bmmonth	c_birth_month
custv.byear	c_birth_year
custv.cust_birth_country	c_birth_country
custv.cust_login_id	c_login
custv.cust_email_address	c_email_address
custv.lreview	c_last_review_date

#### 5.3.13.5 DM\_W

```

CREATE view wrhsv as
SELECT w_warehouse_sk
      ,wrhs_warehouse_id w_warehouse_id
      ,wrhs_warehouse_desc w_warehouse_name
      ,wrhs_warehouse_sq_ft w_warehouse_sq_ft
      ,w_street_number
      ,w_street_name
      ,w_street_type
      ,w_suite_number
      ,w_city
      ,w_county
      ,w_state
      ,w_zip
      ,w_country
      ,w_gmt_offset
FROM   s_warehouse,
       warehouse
WHERE  wrhs_warehouse_id = w_warehouse_id;

```

**Table 5-9: Column mapping for the warehouse dimension**

Source Schema Column	Target Column
Generated as described in generic algorithm	w_warehouse_sk
<b>w_warehouse_id(1)</b>	<b>w_warehouse_id</b>
w_warehouse_name	w_warehouse_name
w_warehouse_sq_ft	w_warehouse_sq_ft
w_street_number	w_street_number
w_street_name	w_street_name
w_street_type	w_street_type
w_suite_number	w_suite_number

Source Schema Column	Target Column
w_city	w_city
w_county	w_county
w_state	w_state
w_zip	w_zip
w_country	w_country
w_gmt_offset	w_gmt_offset

### 5.3.13.6 DM\_CC

```

CREATE view ccv AS
SELECT  next value FOR cc_seq cc_call_center_sk
        ,call_center_id cc_call_center_id
        ,current_date cc_rec_start_date
        ,cast(NULL AS DATE) cc_rec_end_date
        ,d1.d_date_sk cc_closed_date_sk
        ,d2.d_date_sk cc_open_date_sk
        ,call_center_name cc_name
        ,call_center_class cc_class
        ,call_center_employees cc_employees
        ,call_center_sq_ft cc_sq_ft
        ,call_center_hours cc_hours
        ,call_center_manager cc_manager
        ,cc_mkt_id
        ,cc_mkt_class
        ,cc_mkt_desc
        ,cc_market_manager
        ,cc_division
        ,cc_division_name
        ,cc_company
        ,cc_company_name
        ,cc_street_number,cc_street_name,cc_street_type,cc_suite_number,cc_city
        ,cc_county,cc_state,cc_zip
        ,cc_country
        ,cc_gmt_offset
        ,call_center_tax_percentage cc_tax_percentage
FROM s_call_center
LEFT OUTER JOIN date_dim d2 ON d2.d_date = cast(call_closed_date AS DATE)
LEFT OUTER JOIN date_dim d1 ON d1.d_date = cast(call_open_date AS DATE)
LEFT OUTER JOIN call_center ON (call_center_id = cc_call_center_id AND cc_rec_end_date IS NULL);

```

**Table 5-10: Column mapping for the call\_center dimension**

Source Schema Column	Target Column
Generated as described in generic algorithm	cc_call_center_sk
<b>cc_call_center_id(2)</b>	<b>cc_call_center_id</b>
Generated as described in generic algorithm	cc_rec_start_date
Generated as described in generic algorithm	cc_rec_end_date
cc_close_date_sk	cc_close_date_sk
cc_open_date_sk	cc_open_date_sk
cc_name	cc_name
cc_class	cc_class
cc_number_employees	cc_number_employees
cc_square_ft	cc_square_ft
cc_hours	cc_hours
cc_manager	cc_manager
cc_mkt_id	cc_mkt_id
cc_tax_percentage	cc_tax_percentage
cc_mkt_class	cc_mkt_class
cc_mkt_desc	cc_mkt_desc
cc_market_manager	cc_market_manager
cc_division	cc_division
cc_division_name	cc_division_name
cc_company	cc_company
cc_company_name	cc_company_name
cc_street_number	cc_street_number
cc_street_name	cc_street_name
cc_street_type	cc_street_type



Source Schema Column	Target Column
cc_suite_number	cc_suite_number
cc_city	cc_city
cc_county	cc_county
cc_state	cc_state
cc_zip	cc_zip
cc_country	cc_country
cc_gmt_offset	cc_gmt_offset

### 5.3.13.7 DM\_CP

```
CREATE VIEW catv as
SELECT cp_catalog_page_sk
,scp.cpag_id cp_catalog_page_id
,startd.d_date_sk cp_start_date_sk
,endd.d_date_sk cp_end_date_sk
,cpag_department cp_department
,cpag_catalog_number cp_catalog_number
,cpag_catalog_page_number cp_catalog_page_number
,scp.cpag_description cp_description
,scp.cpag_type cp_type
FROM s_catalog_page scp
INNER JOIN date_dim startd ON (scp.cpag_start_date = startd.d_date)
INNER JOIN date_dim endd ON (scp.cpag_end_date = endd.d_date)
INNER JOIN catalog_page cp ON (scp.cpag_id = cp.cp_catalog_page_id);
```

**Table 5-11: Column mapping for the catalog\_page dimension**

Source Schema Column	Target Column
Generated as described in generic algorithm	cp_catalog_page_sk
catv.cpag_catalog_page_number(1)	cp_catalog_page_id
catv.catg_start_date	cp_start_date_sk
catv.catg_end_date	cp_end_date_sk
catv.citm_promotion_id	cp_promo_id
catv.cpag_department	cp_department
catv.cpag_catalog_number	cp_catalog_number
catv.cpag_catalog_page_number	cp_catalog_page_number
catv.catg_catalog_description	cp_description
catv.catg_catalog_type	cp_type

### 5.3.13.8 DM\_P

```
CREATE view promv as
SELECT p_promo_sk
,prom_promotion_id p_promo_id
,d1.d_date_sk p_start_date_sk
,d2.d_date_sk p_end_date_sk
,p_item_sk
,prom_cost p_cost
,prom_response_target p_response_target
,prom_promotion_name p_promo_name
,prom_channel_dmail p_channel_dmail
,prom_channel_email p_channel_email
,prom_channel_catalog p_channel_catalog
,prom_channel_tv p_channel_tv
,prom_channel_radio p_channel_radio
,prom_channel_press p_channel_press
,prom_channel_event p_channel_event
,prom_channel_demo p_channel_demo
,prom_channel_details p_channel_details
,prom_purpose p_purpose
,prom_discount_active p_discount_active
FROM s_promotion
LEFT OUTER JOIN date_dim d1 ON (prom_start_date = d1.d_date)
LEFT OUTER JOIN date_dim d2 ON (prom_end_date = d2.d_date)
LEFT OUTER JOIN promotion ON (prom_promotion_id = p_promo_id);
```

**Table 5-12: Column mapping for the promotion dimension**

Source Schema Column	Target Column
Generated as described in generic orithm	p_promo_sk

Source Schema Column	Target Column
<b>p_promo_id(1)</b>	<b>p_promo_id</b>
p_start_date_sk	p_start_date_sk
p_end_date_sk	p_end_date_sk
p_cost	p_cost
p_response_target	p_response_target
p_promo_name	p_promo_name
p_channel_dmail	p_channel_dmail
p_channel_email	p_channel_email
p_channel_catalog	p_channel_catalog
p_channel_tv	p_channel_tv
p_channel_radio	p_channel_radio
p_channel_press	p_channel_press
p_channel_event	p_channel_event
p_channel_demo	p_channel_demo
p_channel_details	p_channel_details
p_purpose	p_purpose
p_discount_active	p_discount_active

### 5.3.13.9 DM\_WS

```

CREATE VIEW websv as
SELECT next value for web_seq web_site_sk
,wsit_web_site_id web_site_id
,current_date web_rec_start_date
,cast(null as date) web_rec_end_date
,wsit_site_name web_name
,d1.d_date_sk web_open_date_sk
,d2.d_date_sk web_close_date_sk
,wsit_site_class web_class
,wsit_site_manager web_manager
,web_mkt_id
,web_mkt_class
,web_mkt_desc
,web_market_manager
,web_company_id
,web_company_name
,web_street_number
,web_street_name
,web_street_type
,web_suite_number
,web_city
,web_county
,web_state
,web_zip
,web_country
,web_gmt_offset
,wsit_tax_percentage web_tax_percentage
FROM s_web_site
LEFT OUTER JOIN date_dim d1 on (d1.d_date = cast(wsit_open_date as date))
LEFT OUTER JOIN date_dim d2 on (d2.d_date = cast(wsit_closed_date as date))
LEFT OUTER JOIN web_site ON (web_site_id = wsit_web_site_id AND web_rec_end_date is null);

```

**Table 5-13: Column mapping for the web\_site dimension**

Source Schema Column	Target Column
Generated as described in generic algorithm	web_site_sk
<b>web_site_id(2)</b>	web_site_id
Generated as described in generic algorithm	web_rec_start_date
Generated as described in generic algorithm	web_rec_end_date
web_name	web_name
web_open_date_sk	web_open_date_sk
web_close_date_sk	web_close_date_sk
web_class	web_class
web_mgr	web_mgr
web_mkt_id	web_mkt_id
web_mkt_class	web_mkt_class
web_mkt_desc	web_mkt_desc
web_mkt_mgr	web_mkt_mgr

Source Schema Column	Target Column
web_company_id	web_company_id
web_company_name	web_company_name
web_street_number	web_street_number
web_street_name	web_street_name
web_street_type	web_street_type
web_suite_number	web_suite_number
web_city	web_city
web_county	web_county
web_state	web_state
web_zip	web_zip
web_country	web_country
web_gmt_offset	web_gmt_offset

### 5.3.13.10 DM\_WP

```

CREATE view webv as
SELECT next value for wp_seq wp_web_page_sk
,wpag_web_page_id wp_web_page_id
,current_date wp_rec_start_date
,cast(null as date) wp_rec_end_date
,d1.d_date_sk wp_creation_date_sk
,d2.d_date_sk wp_access_date_sk
,wpag_autogen_flag wp_autogen_flag
,wp_customer_sk
,wpag_url wp_url
,wpag_type wp_type
,wpag_char_cnt wp_char_count
,wpag_link_cnt wp_link_count
,wpag_image_cnt wp_image_count
,wpag_max_ad_cnt wp_max_ad_count
FROM s_web_page
LEFT OUTER JOIN date_dim d1 ON cast(wpag_create_date as date) = d1.d_date
LEFT OUTER JOIN date_dim d2 ON cast(wpag_access_date as date) = d2.d_date
LEFT OUTER JOIN web_page ON (wpag_web_page_id = wp_web_page_id AND wp_rec_end_date is null);

```

**Table 5-14: Column mapping for the web\_page dimension**

Source Schema Column	Target Column
Generated as described in generic algorithm	wp_web_page_sk
<b>wp_web_page_id(2)</b>	<b>wp_web_page_id</b>
Generated as described in generic algorithm	wp_rec_start_date
Generated as described in generic algorithm	wp_rec_end_date
wp_creation_date_sk	wp_creation_date_sk
wp_access_date_sk	wp_access_date_sk
wp_autogen_flag	wp_autogen_flag
wp_customer_sk	wp_customer_sk
wp_url	wp_url
wp_type	wp_type
wp_char_count	wp_char_count
wp_link_count	wp_link_count
wp_image_count	wp_image_count
wp_max_ad_count	wp_max_ad_count
wp_tax_percentage	Wp_tax_percentage

### 5.3.13.11 LF\_SS

```
CREATE view ssv as
SELECT  d_date_sk ss_sold_date_sk,
        t_time_sk ss_sold_time_sk,
        i_item_sk ss_item_sk,
        c_customer_sk ss_customer_sk,
        c_current_cdemo_sk ss_cdemo_sk,
        c_current_hdemo_sk ss_hdemo_sk,
        c_current_addr_sk ss_addr_sk,
        s_store_sk ss_store_sk,
        p_promo_sk ss_promo_sk,
        purc_purchase_id ss_ticket_number,
        plin_quantity ss_quantity,
        i_warehouse_cost ss_warehouse_cost,
        i_current_price ss_list_price,
        plin_sale_price ss_sales_price,
        (i_current_price-plin_sale_price)*plin_quantity ss_ext_discount_amt,
        plin_sale_price * plin_quantity ss_ext_sales_price,
        i_warehouse_cost * plin_quantity ss_ext_warehouse_cost,
        i_current_price * plin_quantity ss_ext_list_price,
        i_current_price * s_tax_percentage ss_ext_tax,
        plin_coupon_amt ss_coupon_amt,
        (plin_sale_price * plin_quantity)-plin_coupon_amt ss_net_paid,
        ((plin_sale_price * plin_quantity)-plin_coupon_amt)*(1+s_tax_percentage) ss_net_paid_inc_tax,
        ((plin_sale_price * plin_quantity)-plin_coupon_amt)-(plin_quantity*i_warehouse_cost)
ss_net_profit
FROM      s_purchase
LEFT OUTER JOIN customer ON (purc_customer_id = c_customer_id)
LEFT OUTER JOIN store ON (purc_store_id = s_store_id)
LEFT OUTER JOIN date_dim ON (cast(purc_purchase_date as date) = d_date)
LEFT OUTER JOIN time_dim ON (PURC_PURCHASE_TIME = t_time)
JOIN s_purchase_lineitem ON (purc_purchase_id = plin_purchase_id)
LEFT OUTER JOIN promotion ON plin_promotion_id = p_promo_id
LEFT OUTER JOIN item ON plin_item_id = i_item_id
WHERE     purc_purchase_id = plin_purchase_id
        AND i_rec_end_date is NULL
        AND s_rec_end_date is NULL;
```

**Table 5-15: Column mapping for the store\_sales fact table**

Source Schema Column	Target Column
SS_SOLD_DATE_SK	SS_SOLD_DATE_SK
SS_SOLD_TIME_SK	SS_SOLD_TIME_SK
SS_ITEM_SK	SS_ITEM_SK
SS_CUSTOMER_SK	SS_CUSTOMER_SK
SS_CDEMO_SK	SS_CDEMO_SK
SS_HDEMO_SK	SS_HDEMO_SK
SS_ADDR_SK	SS_ADDR_SK
SS_STORE_SK	SS_STORE_SK
SS_PROMO_SK	SS_PROMO_SK
SS_TICKET_NUMBER	SS_TICKET_NUMBER
SS_QUANTITY	SS_QUANTITY
SS_WHOLESale_COST	SS_WHOLESale_COST
SS_LIST_PRICE	SS_LIST_PRICE
SS_SALES_PRICE	SS_SALES_PRICE
SS_EXT_DISCOUNT_AMT	SS_EXT_DISCOUNT_AMT
SS_EXT_SALES_PRICE	SS_EXT_SALES_PRICE
SS_EXT_WHOLESale_COST	SS_EXT_WHOLESale_COST
SS_EXT_LIST_PRICE	SS_EXT_LIST_PRICE
SS_EXT_TAX	SS_EXT_TAX
SS_COUPON_AMT	SS_COUPON_AMT
SS_NET_PAID	SS_NET_PAID
SS_NET_PAID_INC_TAX	SS_NET_PAID_INC_TAX
SS_SOLD_DATE_SK	SS_NET_PROFIT

### 5.3.13.12 LF\_SR

```

CREATE view srv as
SELECT d_date_sk sr_returned_date_sk
      ,t_time_sk sr_return_time_sk
      ,i_item_sk sr_item_sk
      ,c_customer_sk sr_customer_sk
      ,c_current_demo_sk sr_demo_sk
      ,c_current_hdemo_sk sr_hdemo_sk
      ,c_current_addr_sk sr_addr_sk
      ,s_store_sk sr_store_sk
      ,r_reason_sk sr_reason_sk
      ,sret_ticket_number sr_ticket_number
      ,sret_return_qty sr_return_quantity
      ,sret_return_amt sr_return_amt
      ,sret_return_tax sr_return_tax
      ,sret_return_amt + sret_return_tax sr_return_amt_inc_tax
      ,sret_return_fee sr_fee
      ,sret_return_ship_cost sr_return_ship_cost
      ,sret_refunded_cash sr_refunded_cash
      ,sret_reversed_charge sr_reversed_charge
      ,sret_store_credit sr_store_credit
      ,sret_return_amt+sret_return_tax+sret_return_fee
      -sret_refunded_cash-sret_reversed_charge-sret_store_credit sr_net_loss
FROM s_store_returns
LEFT OUTER JOIN date_dim
  ON (cast(sret_return_date as date) = d_date)
LEFT OUTER JOIN time_dim
  ON (( cast(substr(sret_return_time,1,2) AS integer)*3600
        +cast(substr(sret_return_time,4,2) AS integer)*60
        +cast(substr(sret_return_time,7,2) AS integer)) = t_time)
LEFT OUTER JOIN item ON (sret_item_id = i_item_id)
LEFT OUTER JOIN customer ON (sret_customer_id = c_customer_id)
LEFT OUTER JOIN store ON (sret_store_id = s_store_id)
LEFT OUTER JOIN reason ON (sret_reason_id = r_reason_id)
WHERE i_rec_end_date IS NULL
      AND s_rec_end_date IS NULL;

```

**Table 5-16: Column mapping for the store\_returns fact table**

Source Schema Column	Target Column
SR_RETURNED_DATE_SK	SR_RETURNED_DATE_SK
SR_RETURN_TIME_SK	SR_RETURN_TIME_SK
<b>SR_ITEM_SK</b>	<b>SR_ITEM_SK</b>
SR_CUSTOMER_SK	SR_CUSTOMER_SK
SR_CDEMO_SK	SR_CDEMO_SK
SR_HDEMO_SK	SR_HDEMO_SK
SR_ADDR_SK	SR_ADDR_SK
SR_STORE_SK	SR_STORE_SK
SR_REASON_SK	SR_REASON_SK
<b>SR_TICKET_NUMBER</b>	<b>SR_TICKET_NUMBER</b>
SR_RETURN_QUANTITY	SR_RETURN_QUANTITY
SR_RETURN_AMT	SR_RETURN_AMT
SR_RETURN_TAX	SR_RETURN_TAX
SR_RETURN_AMT_INC_TAX	SR_RETURN_AMT_INC_TAX
SR_FEE	SR_FEE
SR_RETURN_SHIP_COST	SR_RETURN_SHIP_COST
SR_REFUNDED_CASH	SR_REFUNDED_CASH
SR_REVERSED_CHARGE	SR_REVERSED_CHARGE
SR_STORE_CREDIT	SR_STORE_CREDIT
SR_NET_LOSS	SR_NET_LOSS

### 5.3.13.13 LF\_WS

```

CREATE VIEW wsv AS
SELECT  d1.d_date_sk ws_sold_date_sk,
        t_time_sk ws_sold_time_sk,
        d2.d_date_sk ws_ship_date_sk,
        i_item_sk ws_item_sk,
        c1.c_customer_sk ws_bill_customer_sk,
        c1.c_current_cdemo_sk ws_bill_cdemo_sk,
        c1.c_current_hdemo_sk ws_bill_hdemo_sk,
        c1.c_current_addr_sk ws_bill_addr_sk,
        c2.c_customer_sk ws_ship_customer_sk,
        c2.c_current_cdemo_sk ws_ship_cdemo_sk,
        c2.c_current_hdemo_sk ws_ship_hdemo_sk,
        c2.c_current_addr_sk ws_ship_addr_sk,
        wp_web_page_sk ws_web_page_sk,
        web_site_sk ws_web_site_sk,
        sm_ship_mode_sk ws_ship_mode_sk,
        w_warehouse_sk ws_warehouse_sk,
        p_promo_sk ws_promo_sk,
        word_order_id ws_order_number,
        wlin_quantity ws_quantity,
        i_warehouse_cost ws_warehouse_cost,
        i_current_price ws_list_price,
        wlin_sales_price ws_sales_price,
        (i_current_price-wlin_sales_price)*wlin_quantity ws_ext_discount_amt,
        wlin_sales_price * wlin_quantity ws_ext_sales_price,
        i_warehouse_cost * wlin_quantity ws_ext_warehouse_cost,
        i_current_price * wlin_quantity ws_ext_list_price,
        i_current_price * web_tax_percentage ws_ext_tax,
        wlin_coupon_amt ws_coupon_amt,
        wlin_ship_cost * wlin_quantity WS_EXT_SHIP_COST,
        (wlin_sales_price * wlin_quantity)-wlin_coupon_amt ws_net_paid,
        ((wlin_sales_price * wlin_quantity)-wlin_coupon_amt)*(1+web_tax_percentage) ws_net_paid_inc_tax,
        ((wlin_sales_price * wlin_quantity)-wlin_coupon_amt)-(wlin_quantity*i_warehouse_cost)
WS_NET_PAID_INC_SHIP,
        (wlin_sales_price * wlin_quantity)-wlin_coupon_amt + (wlin_ship_cost * wlin_quantity)
        + i_current_price * web_tax_percentage WS_NET_PAID_INC_SHIP_TAX,
        ((wlin_sales_price * wlin_quantity)-wlin_coupon_amt)-(i_warehouse_cost * wlin_quantity)
WS_NET_PROFIT
FROM      s_web_order
LEFT OUTER JOIN date_dim d1 ON (cast(word_order_date as date) = d1.d_date)
LEFT OUTER JOIN time_dim ON (word_order_time = t_time)
LEFT OUTER JOIN customer c1 ON (word_bill_customer_id = c1.c_customer_id)
LEFT OUTER JOIN customer c2 ON (word_ship_customer_id = c2.c_customer_id)
LEFT OUTER JOIN web_site ON (word_web_site_id = web_site_id AND web_rec_end_date IS NULL)
LEFT OUTER JOIN ship_mode ON (word_ship_mode_id = sm_ship_mode_id)
JOIN s_web_order_lineitem ON (word_order_id = wlin_order_id)
LEFT OUTER JOIN date_dim d2 ON (cast(wlin_ship_date as date) = d2.d_date)
LEFT OUTER JOIN item ON (wlin_item_id = i_item_id AND i_rec_end_date IS NULL)
LEFT OUTER JOIN web_page ON (wlin_web_page_id = wp_web_page_id AND wp_rec_end_date IS NULL)
LEFT OUTER JOIN warehouse ON (wlin_warehouse_id = w_warehouse_id)
LEFT OUTER JOIN promotion ON (wlin_promotion_id = p_promo_id);

```

**Table 5-17: Column mapping for the web\_sales fact table**

Source Schema Column	Target Column
WS_SOLD_DATE_SK	WS_SOLD_DATE_SK
WS_SOLD_TIME_SK	WS_SOLD_TIME_SK
WS_SHIP_DATE_SK	WS_SHIP_DATE_SK
WS_ITEM_SK	WS_ITEM_SK
WS_BILL_CUSTOMER_SK	WS_BILL_CUSTOMER_SK
WS_BILL_CDEMO_SK	WS_BILL_CDEMO_SK
WS_BILL_HDEMO_SK	WS_BILL_HDEMO_SK
WS_BILL_ADDR_SK	WS_BILL_ADDR_SK
WS_SHIP_CUSTOMER_SK	WS_SHIP_CUSTOMER_SK
WS_SHIP_CDEMO_SK	WS_SHIP_CDEMO_SK
WS_SHIP_HDEMO_SK	WS_SHIP_HDEMO_SK
WS_SHIP_ADDR_SK	WS_SHIP_ADDR_SK
WS_WEB_PAGE_SK	WS_WEB_PAGE_SK

Source Schema Column	Target Column
WS_WEB_SITE_SK	WS_WEB_SITE_SK
WS_SHIP_MODE_SK	WS_SHIP_MODE_SK
WS_WAREHOUSE_SK	WS_WAREHOUSE_SK
WS_PROMO_SK	WS_PROMO_SK
<b>WS_ORDER_NUMBER</b>	<b>WS_ORDER_NUMBER</b>
WS_QUANTITY	WS_QUANTITY
WS_WHOLESALE_COST	WS_WHOLESALE_COST
WS_LIST_PRICE	WS_LIST_PRICE
WS_SALES_PRICE	WS_SALES_PRICE
WS_EXT_DISCOUNT_AMT	WS_EXT_DISCOUNT_AMT
WS_EXT_SALES_PRICE	WS_EXT_SALES_PRICE
WS_EXT_WHOLESALE_COST	WS_EXT_WHOLESALE_COST
WS_EXT_LIST_PRICE	WS_EXT_LIST_PRICE
WS_EXT_TAX	WS_EXT_TAX
WS_COUPON_AMT	WS_COUPON_AMT
WS_EXT_SHIP_COST	WS_EXT_SHIP_COST
WS_NET_PAID	WS_NET_PAID
WS_NET_PAID_INC_TAX	WS_NET_PAID_INC_TAX
WS_NET_PAID_INC_SHIP	WS_NET_PAID_INC_SHIP
WS_NET_PAID_INC_SHIP_TAX	WS_NET_PAID_INC_SHIP_TAX
WS_NET_PROFIT	WS_NET_PROFIT

### 5.3.13.14 LF\_WR

```
CREATE VIEW wrv AS
SELECT d_date_sk wr_return_date_sk
      ,t_time_sk wr_return_time_sk
      ,i_item_sk wr_item_sk
      ,c1.c_customer_sk wr_refunded_customer_sk
      ,c1.c_current_cdemo_sk wr_refunded_cdemo_sk
      ,c1.c_current_hdemo_sk wr_refunded_hdemo_sk
      ,c1.c_current_addr_sk wr_refunded_addr_sk
      ,c2.c_customer_sk wr_returning_customer_sk
      ,c2.c_current_cdemo_sk wr_returning_cdemo_sk
      ,c2.c_current_hdemo_sk wr_returning_hdemo_sk
      ,c2.c_current_addr_sk wr_returning_addr_sk
      ,wp_web_page_sk wr_web_page_sk
      ,r_reason_sk wr_reason_sk
      ,wret_order_id wr_order_number
      ,wret_return_qty wr_return_quantity
      ,wret_return_amt wr_return_amt
      ,wret_return_tax wr_return_tax
      ,wret_return_amt + wret_return_tax AS wr_return_amt_inc_tax
      ,wret_return_fee wr_fee
      ,wret_return_ship_cost wr_return_ship_cost
      ,wret_refunded_cash wr_refunded_cash
      ,wret_reversed_charge wr_reversed_charge
      ,wret_account_credit wr_account_credit
      ,wret_return_amt+wret_return_tax+wret_return_fee
      -wret_refunded_cash-wret_reversed_charge-wret_account_credit wr_net_loss
FROM s_web_returns LEFT OUTER JOIN date_dim ON (cast(wret_return_date as date) = d_date)
LEFT OUTER JOIN time_dim ON ((CAST(SUBSTR(wret_return_time,1,2) AS integer)*3600
+CAST(SUBSTR(wret_return_time,4,2) AS integer)*60+CAST(SUBSTR(wret_return_time,7,2) AS integer))=t_time)
LEFT OUTER JOIN item ON (wret_item_id = i_item_id)
LEFT OUTER JOIN customer c1 ON (wret_return_customer_id = c1.c_customer_id)
LEFT OUTER JOIN customer c2 ON (wret_refund_customer_id = c2.c_customer_id)
LEFT OUTER JOIN reason ON (wret_reason_id = r_reason_id)
LEFT OUTER JOIN web_page ON (wret_web_page_id = WP_WEB_PAGE_id)
WHERE i_rec_end_date IS NULL AND wp_rec_end_date IS NULL;
```

**Table: 5-18: Column mapping for the web\_returns fact table**

Source Schema Column	Target Column
WR_RETURNED_DATE_SK	WR_RETURNED_DATE_SK
WR_RETURNED_TIME_SK	WR_RETURNED_TIME_SK
WR_SHIP_DATE_SK	WR_SHIP_DATE_SK
<b>WR_ITEM_SK</b>	<b>WR_ITEM_SK</b>
WR_REFUNDED_CUSTOMER_SK	WR_REFUNDED_CUSTOMER_SK
WR_REFUNDED_CDEMO_SK	WR_REFUNDED_CDEMO_SK
WR_REFUNDED_HDEMO_SK	WR_REFUNDED_HDEMO_SK
WR_REFUNDED_ADDR_SK	WR_REFUNDED_ADDR_SK
WR_RETURNING_CUSTOMER_SK	WR_RETURNING_CUSTOMER_SK
WR_RETURNING_CDEMO_SK	WR_RETURNING_CDEMO_SK
WR_RETURNING_HDEMO_SK	WR_RETURNING_HDEMO_SK
WR_RETURNING_ADDR_SK	WR_RETURNING_ADDR_SK
WR_WEB_PAGE_SK	WR_WEB_PAGE_SK
WR_SHIP_MODE_SK	WR_SHIP_MODE_SK
WR_REASON_SK	WR_REASON_SK
WR_WAREHOUSE_SK	WR_WAREHOUSE_SK
<b>WR_ORDER_NUMBER</b>	<b>WR_ORDER_NUMBER</b>
WR_RETURN_QUANTITY	WR_RETURN_QUANTITY
WR_RETURN_AMT	WR_RETURN_AMT
WR_RETURN_TAX	WR_RETURN_TAX
WR_RETURN_AMT_INC_TAX	WR_RETURN_AMT_INC_TAX
WR_FEE	WR_FEE
WR_RETURN_SHIP_COST	WR_RETURN_SHIP_COST
WR_REFUNDED_CASH	WR_REFUNDED_CASH
WR_REVERSED_CHARGE	WR_REVERSED_CHARGE
WR_ACCOUNT_CREDIT	WR_ACCOUNT_CREDIT
WR_NET_LOSS	WR_NET_LOSS



### 5.3.13.15 LF\_CS

```
CREATE view csv as
SELECT d1.d_date_sk cs_sold_date_sk
, t_time_sk cs_sold_time_sk
, d2.d_date_sk cs_ship_date_sk
, c1.c_customer_sk cs_bill_customer_sk
, c1.c_current_cdemo_sk cs_bill_cdemo_sk
, c1.c_current_hdemo_sk cs_bill_hdemo_sk
, c1.c_current_addr_sk cs_bill_addr_sk
, c2.c_customer_sk cs_ship_customer_sk
, c2.c_current_cdemo_sk cs_ship_cdemo_sk
, c2.c_current_hdemo_sk cs_ship_hdemo_sk
, c2.c_current_addr_sk cs_ship_addr_sk
, cc.call_center_sk cs_call_center_sk
, cp.catalog_page_sk cs_catalog_page_sk
, sm.ship_mode_sk cs_ship_mode_sk
, w_warehouse_sk cs_warehouse_sk
, i_item_sk cs_item_sk
, p_promo_sk cs_promo_sk
, cord_order_id cs_order_number
, clin_quantity cs_quantity
, i_warehouse_cost cs_warehouse_cost
, i_current_price cs_list_price
, clin_sales_price cs_sales_price
, (i_current_price-clin_sales_price)*clin_quantity cs_ext_discount_amt
, clin_sales_price * clin_quantity cs_ext_sales_price
, i_warehouse_cost * clin_quantity cs_ext_warehouse_cost
, i_current_price * clin_quantity CS_EXT_LIST_PRICE
, i_current_price * cc_tax_percentage CS_EXT_TAX
, clin_coupon_amt cs_coupon_amt
, clin_ship_cost * clin_quantity CS_EXT_SHIP_COST
, (clin_sales_price * clin_quantity)-clin_coupon_amt cs_net_paid
, ((clin_sales_price * clin_quantity)-clin_coupon_amt)*(1+cc_tax_percentage) cs_net_paid_inc_tax
, (clin_sales_price * clin_quantity)-clin_coupon_amt + (clin_ship_cost * clin_quantity) CS_NET_PAID_INC_SHIP
, (clin_sales_price * clin_quantity)-clin_coupon_amt + (clin_ship_cost * clin_quantity)
+ i_current_price * cc_tax_percentage CS_NET_PAID_INC_SHIP_TAX
, ((clin_sales_price * clin_quantity)-clin_coupon_amt)-(clin_quantity*i_warehouse_cost) cs_net_profit
FROM s_catalog_order
LEFT OUTER JOIN date_dim d1 ON
    (cast(cord_order_date as date) = d1.d_date)
LEFT OUTER JOIN time_dim ON (cord_order_time = t_time)
LEFT OUTER JOIN customer c1 ON (cord_bill_customer_id = c1.c_customer_id)
LEFT OUTER JOIN customer c2 ON (cord_ship_customer_id = c2.c_customer_id)
LEFT OUTER JOIN call_center ON (cord_call_center_id = cc.call_center_id AND cc_rec_end_date IS NULL)
LEFT OUTER JOIN ship_mode ON (cord_ship_mode_id = sm.ship_mode_id)
JOIN s_catalog_order_lineitem ON (cord_order_id = clin_order_id)
LEFT OUTER JOIN date_dim d2 ON
    (cast(clin_ship_date as date) = d2.d_date)
LEFT OUTER JOIN catalog_page ON
    (clin_catalog_page_number = cp.catalog_page_number and clin_catalog_number = cp_catalog_number)
LEFT OUTER JOIN warehouse ON (clin_warehouse_id = w_warehouse_id)
LEFT OUTER JOIN item ON (clin_item_id = i_item_id AND i_rec_end_date IS NULL)
LEFT OUTER JOIN promotion ON (clin_promotion_id = p_promo_id);
```

**Table 5-19: Column mapping for the catalog\_sales fact table**

Source Schema Column	Target Column
CS_SOLD_DATE_SK	CS_SOLD_DATE_SK
CS_SOLD_TIME_SK	CS_SOLD_TIME_SK
CS_SHIP_DATE_SK	CS_SHIP_DATE_SK
CS_BILL_CUSTOMER_SK	CS_BILL_CUSTOMER_SK
CS_BILL_CDEMO_SK	CS_BILL_CDEMO_SK
CS_BILL_HDEMO_SK	CS_BILL_HDEMO_SK
CS_BILL_ADDR_SK	CS_BILL_ADDR_SK
CS_SHIP_CUSTOMER_SK	CS_SHIP_CUSTOMER_SK
CS_SHIP_CDEMO_SK	CS_SHIP_CDEMO_SK
CS_SHIP_HDEMO_SK	CS_SHIP_HDEMO_SK
CS_SHIP_ADDR_SK	CS_SHIP_ADDR_SK
CS_CALL_CENTER_SK	CS_CALL_CENTER_SK
CS_CATALOG_PAGE_SK	CS_CATALOG_PAGE_SK
CS_SHIP_MODE_SK	CS_SHIP_MODE_SK
CS_WAREHOUSE_SK	CS_WAREHOUSE_SK
CS_ITEM_SK	CS_ITEM_SK
CS_PROMO_SK	CS_PROMO_SK
CS_ORDER_NUMBER	CS_ORDER_NUMBER

Source Schema Column	Target Column
CS_QUANTITY	CS_QUANTITY
CS_WHOLESALE_COST	CS_WHOLESALE_COST
CS_LIST_PRICE	CS_LIST_PRICE
CS_SALES_PRICE	CS_SALES_PRICE
CS_EXT_DISCOUNT_AMT	CS_EXT_DISCOUNT_AMT
CS_EXT_SALES_PRICE	CS_EXT_SALES_PRICE
CS_EXT_WHOLESALE_COST	CS_EXT_WHOLESALE_COST
CS_EXT_LIST_PRICE	CS_EXT_LIST_PRICE
CS_EXT_TAX	CS_EXT_TAX
CS_COUPON_AMT	CS_COUPON_AMT
CS_EXT_SHIP_COST	CS_EXT_SHIP_COST
CS_NET_PAID	CS_NET_PAID
CS_NET_PAID_INC_TAX	CS_NET_PAID_INC_TAX
CS_NET_PAID_INC_SHIP	CS_NET_PAID_INC_SHIP
CS_NET_PAID_INC_SHIP_TAX	CS_NET_PAID_INC_SHIP_TAX
CS_NET_PROFIT	CS_NET_PROFIT

### 5.3.13.16 LF\_CR

```
CREATE VIEW crv as
SELECT d_date_sk cr_return_date_sk
      ,t_time_sk cr_return_time_sk
      ,i_item_sk cr_item_sk
      ,c1.c_customer_sk cr_refunded_customer_sk
      ,c1.c_current_cdemo_sk cr_refunded_cdemo_sk
      ,c1.c_current_hdemo_sk cr_refunded_hdemo_sk
      ,c1.c_current_addr_sk cr_refunded_addr_sk
      ,c2.c_customer_sk cr_returning_customer_sk
      ,c2.c_current_cdemo_sk cr_returning_cdemo_sk
      ,c2.c_current_hdemo_sk cr_returning_hdemo_sk
      ,c2.c_current_addr_sk cr_returning_addr_sk
      ,cc_call_center_sk cr_call_center_sk
      ,cp_catalog_page_sk CR_CATALOG_PAGE_SK
      ,sm_ship_mode_sk CR_SHIP_MODE_SK
      ,w_warehouse_sk CR_WAREHOUSE_SK
      ,r_reason_sk cr_reason_sk
      ,cret_order_id cr_order_number
      ,cret_return_qty cr_return_quantity
      ,cret_return_amt cr_return_amt
      ,cret_return_tax cr_return_tax
      ,cret_return_amt + cret_return_tax AS cr_return_amt_inc_tax
      ,cret_return_fee cr_fee
      ,cret_return_ship_cost cr_return_ship_cost
      ,cret_refunded_cash cr_refunded_cash
      ,cret_reversed_charge cr_reversed_charge
      ,cret_merchant_credit cr_merchant_credit
      ,cret_return_amt+cret_return_tax+cret_return_fee
      -cret_refunded_cash-cret_reversed_charge-cret_merchant_credit cr_net_loss
FROM s_catalog_returns
LEFT OUTER JOIN date_dim
  ON (cast(cret_return_date as date) = d_date)
LEFT OUTER JOIN time_dim ON
  ((CAST(substr(cret_return_time,1,2) AS integer)*3600
    +CAST(substr(cret_return_time,4,2) AS integer)*60
    +CAST(substr(cret_return_time,7,2) AS integer)) = t_time)
LEFT OUTER JOIN item ON (cret_item_id = i_item_id)
LEFT OUTER JOIN customer c1 ON (cret_return_customer_id = c1.c_customer_id)
LEFT OUTER JOIN customer c2 ON (cret_refund_customer_id = c2.c_customer_id)
LEFT OUTER JOIN reason ON (cret_reason_id = r_reason_id)
LEFT OUTER JOIN call_center ON (cret_call_center_id = cc_call_center_id)
LEFT OUTER JOIN catalog_page ON (cret_catalog_page_id = cp_catalog_page_id)
LEFT OUTER JOIN ship_mode ON (cret_shipmode_id = sm_ship_mode_id)
LEFT OUTER JOIN warehouse ON (cret_warehouse_id = w_warehouse_id)
WHERE i_rec_end_date IS NULL AND cc_rec_end_date IS NULL;
```

**Table 5-20: Column mapping for the catalog\_returns fact table**

Source Schema Column	Target Column
CR_RETURNED_DATE_SK	CR_RETURNED_DATE_SK
CR_RETURNED_TIME_SK	CR_RETURNED_TIME_SK
CR_SHIP_DATE_SK	CR_SHIP_DATE_SK
<b>CR_ITEM_SK</b>	<b>CR_ITEM_SK</b>
CR_REFUNDED_CUSTOMER_SK	CR_REFUNDED_CUSTOMER_SK
CR_REFUNDED_CDEMO_SK	CR_REFUNDED_CDEMO_SK
CR_REFUNDED_HDEMO_SK	CR_REFUNDED_HDEMO_SK
CR_REFUNDED_ADDR_SK	CR_REFUNDED_ADDR_SK
CR_RETURNING_CUSTOMER_SK	CR_RETURNING_CUSTOMER_SK
CR_RETURNING_CDEMO_SK	CR_RETURNING_CDEMO_SK
CR_RETURNING_HDEMO_SK	CR_RETURNING_HDEMO_SK
CR_RETURNING_ADDR_SK	CR_RETURNING_ADDR_SK
CR_CALL_CENTER_SK	CR_CALL_CENTER_SK
CR_CATALOG_PAGE_SK	CR_CATALOG_PAGE_SK
CR_SHIP_MODE_SK	CR_SHIP_MODE_SK
CR_WAREHOUSE_SK	CR_WAREHOUSE_SK
CR_REASON_SK	CR_REASON_SK
<b>CR_ORDER_NUMBER</b>	<b>CR_ORDER_NUMBER</b>

Source Schema Column	Target Column
CR_RETURN_QUANTITY	CR_RETURN_QUANTITY
CR_RETURN_AMOUNT	CR_RETURN_AMOUNT
CR_RETURN_TAX	CR_RETURN_TAX
CR_RETURN_AMT_INC_TAX	CR_RETURN_AMT_INC_TAX
CR_FEE	CR_FEE
CR_RETURN_SHIP_COST	CR_RETURN_SHIP_COST
CR_REFUNDED_CASH	CR_REFUNDED_CASH
CR_REVERSED_CHARGE	CR_REVERSED_CHARGE
CR_ACCOUNT_CREDIT	CR_ACCOUNT_CREDIT
CR_NET_LOSS	CR_NET_LOSS

5.3.13.17 LF\_I:

5.3.13.18

```
CREATE view iv AS
SELECT d_date_sk inv_date_sk,
       i_item_sk inv_item_sk,
       w_warehouse_sk inv_warehouse_sk,
       invn_qty_on_hand inv_quantity_on_hand
FROM s_inventory
LEFT OUTER JOIN warehouse ON (invn_warehouse_id=w_warehouse_id)
LEFT OUTER JOIN item ON (invn_item_id=i_item_id AND i_rec_end_date IS NULL)
LEFT OUTER JOIN date_dim ON (d_date=invn_date);
```

**Table 5-21: Column mapping for the inventory fact table**

Source Schema Column	Target Column
inv_date_sk	inv_date_sk
inv_item_sk	inv_item_sk
inv_warehouse_sk	inv_warehouse_sk
inv_quantity_on_hand	inv_quantity_on_hand

5.3.13.19 DF\_SS:

S=store\_sales  
R=store\_returns  
Date1 as generated by dsdgen  
Date2 as generated by dsdgen

5.3.13.20 DF\_CS:

S=catalog\_sales  
R=catalog\_returns  
Date1 as generated by dsdgen  
Date2 as generated by dsdgen

5.3.13.21 DF\_WS:

S=web\_sales  
R=web\_returns  
Date1 as generated by dsdgen  
Date2 as generated by dsdgen

5.3.13.22 DF\_I:

I=Inventory  
Date1 as generated by dsdgen  
Date2 as generated by dsdgen

## 6 Data Persistency Properties

### 6.1 The ACID Properties

The ACID (Atomicity, Consistency, Isolation, and Durability) properties of transaction processing systems shall be supported by the system under test (SUT) during the timed portion of this benchmark. The ACID properties are evaluated outside the timed portion of the test. It is the intent of this section to informally define the ACID properties and to specify a series of tests that can be performed to demonstrate that these properties are met.

- 6.1.1 While the SUT is required to support the ACID properties defined in this clause, the execution of the corresponding ACID tests is only required in lieu of supplying sufficient evidence of the SUT's support for these ACID properties.

**Comment:** No finite series of tests can prove that the ACID properties are fully supported. Being able to pass the specified tests is a necessary, but not sufficient, condition for meeting the ACID requirements.

**Comment:** The ACID tests are intended to demonstrate that the ACID properties are supported by the SUT and enabled during the performance measurements. They are not intended to be an exhaustive quality assurance test.

#### 6.1.2 General Constraints

- 6.1.2.1 The ACID tests must be performed against the qualification database (see Clause 3.3.1). The same set of mechanisms enabled and used to ensure full ACID properties of the qualification database during the ACID tests must be enabled and used for the test database during the performance test. This applies both to attributes of the database (including tables and auxiliary structures) and to attributes of the database session(s) used to execute the ACID and performance tests. The attributes of the session executing the ACID Query (see Clause 6.1.5.3) must be the same as those used in the performance test query sessions(s), and the attributes of the session executing the ACID transaction (see Clause 6.1.5.2) must be the same as those used in the refresh sets (see Clause 7.4.9).

- 6.1.2.2 The term “attributes” includes all database properties and characteristics that can be externally defined. Configuration and initialization files, environmental settings, SQL commands, stored procedures, loadable modules and plug-ins all have the potential to alter or modify database properties and characteristics, and must be considered in assessing compliance with Clause 6.1.2.1.

- 6.1.2.3 ACID properties must be supported for all tables and EADS of the TPC-DS database.

- 6.1.3 All mechanisms used to ensure durability of the qualification database must be enabled and used for the test database. For example:

- a) If the qualification database relies on undo logs to ensure atomicity, then the same logging must also be enabled for the test database during the performance test.
- b) If the qualification database relies on a database backup to meet the durability requirement (see Clause 6.5), a backup must be taken of the test database.
- c) If the qualification database relies on device protection mechanisms (e.g. RAID or mirroring) to meet the durability requirement (see Clause 6.5), these mechanisms must be active during the execution of the performance test.

- 6.1.4 The test sponsor must attest that the reported configuration would also pass the ACID tests with the test database.

#### 6.1.5 The ACID Transaction and the ACID Query

- 6.1.5.1 Since this benchmark does not contain any OLTP transaction, a special ACID Transaction is defined for use in some of the ACID tests. In addition, to simplify the demonstration that ACID properties are enabled while read-only queries are executing concurrently with other activities, a special ACID Query is defined.

- 6.1.5.2 The ACID Transaction is based on the data maintenance function for the ITEM table. It must be implemented using the following components from the data maintenance operations:
- a) Run dsdgen to generate the flat files needed to refresh the item table in the qualification database with a new update set.
  - b) Populate the data warehouse source tables for the qualification database as defined in the data maintenance section.
  - c) Execute the ITEM maintenance transaction as defined in Clause 5.3.13.2.

**Comment:** It is the implementer's responsibility to assure that the ACID transaction runs on a new dataset every time it is invoked.

- 6.1.5.3 The ACID Query must be implemented to conform to the following functional query definition:

```
SELECT *  
FROM item WHERE i_item_id in (SELECT item_item_id  
                             FROM s_item)  
ORDER BY i_item_id;
```

## 6.2 Atomicity Requirements

### 6.2.1 Atomicity Property Definition

The system under test must guarantee that transactions are atomic. The system will either perform all individual operations on the data, or will assure that no partially completed operations leave any effects on the data.

### 6.2.2 Atomicity Tests

1. Run the ACID query and write the results to file1.
2. Perform the ACID Transaction and COMMIT.
3. Run the ACID query and write the results to file2.
4. Compare file1 and file2 to verify that the appropriate rows have been changed in the ITEM table.
5. Perform the ACID Transaction, substituting a ROLLBACK of the transaction for the COMMIT of the transaction.
6. Verify that the appropriate rows have not been changed in the item table.
7. Run the ACID query and write the results to file3.
8. Compare file2 and file3 to verify that no rows have been changed in the ITEM table.

## 6.3 Consistency Requirements

### 6.3.1 Consistency Property Definition

Consistency is the property of the application that requires any execution of transactions to take the database from one consistent state to another.

### 6.3.2 Consistency Condition

- 6.3.2.1 A consistent state for the TPC-DS database is defined to exist when the following query returns a count of 0.

```

SELECT count(*)
FROM
  (SELECT i_item_id,
         sum(case WHEN(i_rec_end_date is null) then 1 else 0 end) as cnt
    FROM item
   GROUP BY i_item_id
   HAVING cnt <> 1
  ) ctab;

```

**Comment:** If this query returns a value of 0 , it shows that all items have one and exactly one ‘current’ value where i\_rec\_end\_date is NULL.

The database is consistent if the returned count is zero.

6.3.2.2 A TPC-DS database, when populated as defined in Clause 3 must meet the consistency condition defined in Clause 6.3.2.

6.3.2.3 If data is replicated, as permitted under Clause **Error! Reference source not found.**, each copy must meet the consistency condition defined in Clause 6.3.2.1.

### 6.3.3 Consistency Tests

To verify the consistency within the ITEM table, perform the following steps:

- Verify that the ITEM table is initially consistent as defined in Clause 6.3.2.1.
- Start at least  $S_q+1$  sessions, with  $S_q$  being the number of sessions used in the Query Run (see Clause 7.4.7), and submit a single ACID Transaction in each session.
- Re-verify the consistency of the ITEM table as defined in Clause 6.3.2.1.

## 6.4 Isolation Requirements

### 6.4.1 Isolation Property Definition

Isolation can be defined in terms of the following phenomena that may occur during the execution of concurrent database transactions (i.e., read-write transactions or read-only queries):

- P0 ("Dirty Write"): Database transaction T1 reads a data element and modifies it. Database transaction T2 then modifies or deletes that data element, and performs a COMMIT. If T1 were to attempt to re-read the data element, it may receive the modified value from T2 or discover that the data element has been deleted.
- P1 ("Dirty Read"): Database transaction T1 modifies a data element. Database transaction T2 then reads that data element before T1 performs a COMMIT. If T1 were to perform a ROLLBACK, T2 will have read a value that was never committed and that may thus be considered to have never existed.
- P2 ("Non-repeatable Read"): Database transaction T1 reads a data element. Database transaction T2 then modifies or deletes that data element, and performs a COMMIT. If T1 were to attempt to re-read the data element, it may receive the modified value or discover that the data element has been deleted.
- P3 ("Phantom"): Database transaction T1 reads a set of values N that satisfy some <search condition>. Database transaction T2 then executes statements that generate one or more data elements that satisfy the <search condition> used by database transaction T1. If database transaction T1 were to repeat the initial read with the same <search condition>, it obtains a different set of values.

Each database transaction T1 and T2 above must be executed completely or not at all.

The following table defines four isolation levels with respect to the phenomena P0, P1, P2, and P3.

**Table 6-1 Isolation Levels**

	Phenomenon P0	Phenomenon P1	Phenomenon P2	Phenomenon P3
Level 0	Not Possible	Possible	Possible	Possible
Level 1	Not Possible	Not Possible	Possible	Possible
Level 2	Not Possible	Not Possible	Not Possible	Possible
Level 3	Not Possible	Not Possible	Not Possible	Not Possible

The following terms are defined:

$T_1$  = An instance of the ACID Transaction;

$T_2$  = An instance of the ACID Transaction;

$T_3$  = Any of the TPC-DS queries or an instance of the ACID query;

$T_n$  = Any arbitrary transaction.

Although arbitrary, the transaction  $T_n$  shall not do dirty writes.

The following table defines the isolation requirements that shall be met by TPC-DS implementations

**Table 6-2 Isolation Requirements**

Req. #	For transactions in this set:	these phenomena:	must NOT be seen by this transaction:	Textual Description:
1.	$\{T_i, T_j\} 1 \leq i, j \leq 2$	P0, P1, P2, P3	$T_i$	Level 3 isolation between any two ACID Transactions.
2.	$\{T_i, T_n\} 1 \leq i \leq 2$	P0, P1, P2	$T_i$	Level 2 isolation for any ACID Transaction relative to any arbitrary transaction.
3.	$\{T_i, T_3\} 1 \leq i \leq n$	P0, P1	$T_3$	Level 1 isolation for any of TPC-DS queries relative to any ACID Transaction and any arbitrary transaction.

Sufficient conditions shall be enabled at either the system or application level to ensure the required isolation, as defined above, is obtained.

The required isolation levels shall not be obtained by the use of configurations or explicit session-level options that give a particular session or transaction *a priori* exclusive access to the database.

The intent is not to preclude automatic mechanisms such as lock escalation, but to disallow configurations and options that would *a priori* preclude queries and update transactions against the same database from making progress concurrently.

In addition, the configuration of the database or session-level options must be such that the continuous submission of arbitrary (read-only) queries against one or more tables could not indefinitely delay update transactions affecting those tables from making progress.

#### 6.4.2 Isolation Tests



#### 6.4.2.1 General Requirements

6.4.2.2 Isolation may be tested as described below. Systems that implement an isolation scheme that does not result in the behavior described in these isolation tests may require different validation techniques. It is permissible to implement the isolation tests differently than as described below. If different tests and techniques are used, they must be fully disclosed by the test sponsor.

The six tests described here are designed to verify that the system under test is configured to support the required isolation levels, as defined in Table 6-2 Isolation Requirements.

##### Isolation Test 1

This test demonstrates isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is committed. Perform the following steps:

1. Run the ACID Query and write the results to file1.
2. Start an ACID Transaction Txn1.
3. Suspend Txn1 immediately prior to COMMIT.
4. Run the ACID Query and write results to file2 (it attempts to read the data that has just been updated by Txn1.)
5. Allow Txn1 to complete.
6. The ACID Query should now have completed.
7. Compare file1 and file2 to verify that the ACID query does not see Txn1's updates.

#### 6.4.2.3 Isolation Test 2

This test demonstrates isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is rolled back. Perform the following steps:

1. Run the ACID query and write the results to file1.
2. Start an ACID Transaction Txn1.
3. Suspend Txn1 immediately prior to COMMIT.
4. Run the ACID Query and write the results to file2 (it attempts to read the data that has just been updated by Txn1.)
5. Force Txn1 to rollback.
6. The ACID Query should now have completed.
7. Compare file1 and file2 to verify that the ACID query does not see Txn1's updates.

#### 6.4.2.4 Isolation Test 3

This test demonstrates isolation for the write-write conflict of two update transactions when the first transaction is committed. Perform the following steps:

1. Run the ACID Query and write the results to file1.
2. Start an ACID Transaction Txn1.
3. Stop Txn1 immediately prior to COMMIT.
4. Start another ACID Transaction Txn2 (Txn2 attempts to read and update the data that has just been updated by Txn1.)
5. Verify that Txn2 waits.
6. Allow Txn1 to complete. Txn2 should now complete.
7. Run the ACID Query and write the results to file2.
8. Compare file1 and file2 to verify that the selected items contain the correct end dates from Txn2.

#### 6.4.2.5 Isolation Test 4

This test demonstrates isolation for the write-write conflict of two update transactions when the first transaction is rolled back. Perform the following steps:

1. Run the ACID Query and write the results to file1.
2. Start an ACID Transaction Txn1.
3. Stop Txn1 immediately prior to COMMIT.
4. Start another ACID Transaction Txn2 (Txn2 attempts to read and update the data that has just been updated by Txn1.)
5. Verify that Txn2 waits.
6. Force Txn1 to rollback. Txn2 should now complete.
7. Run the ACID Query and write the results to file2.
8. Compare file1 and file2 to verify that the selected items contain the correct end dates from Txn2.

#### 6.4.2.6 Isolation Test 5

This test demonstrates the ability of read and write transactions affecting different database tables to make progress concurrently.

1. Run the ACID Query and write the results to file1.
2. Start an ACID Transaction Txn1.
3. Suspend Txn1 immediately prior to COMMIT.
4. Start a transaction Txn2 that does the following:

Select random value of S\_KEY. Return all columns of the STORE table for which S\_STORE\_KEY is equal to the selected value.

1. Verify that Txn2 completes.
2. Allow Txn1 to complete.
3. Run the ACID Query and write the results to file2.
4. Compare file1 and file2 to verify that the appropriate rows in the ITEM table have been changed.

#### 6.4.2.7 Isolation Test 6

This test demonstrates that the continuous submission of arbitrary (read-only) queries against one or more tables of the database does not indefinitely delay update transactions affecting those queries from making progress.

1. Run the ACID Query and write the results to file1.
2. Start a transaction Txn1. Txn1 executes ACIDQ1 against the qualification database so that the Query Runs for a sufficient length of time.

ACIDQ1:

```
select count(ss_sold_date_sk)
      ,count(ss_sold_time_sk),count(ss_item_sk)
      ,count(ss_customer_sk),count(ss_demo_sk)
      ,count(ss_hdemo_sk),count(ss_addr_sk)
      ,count(ss_store_sk),count(ss_promo_sk)
      ,count(ss_ticket_number),count(ss_quantity)
      ,count(ss_wholesale_cost),count(ss_list_price)
      ,count(ss_sales_price),count(ss_ext_discount_amt)
      ,count(ss_ext_sales_price),count(ss_ext_wholesale_cost)
      ,count(ss_ext_list_price),count(ss_ext_tax)
      ,count(ss_coupon_amt),count(ss_net_paid)
      ,count(ss_net_paid_inc_tax),count(ss_net_profit)
      ,count(i_item_id),count(i_rec_start_date)
      ,count(i_rec_end_date),count(i_item_desc)
      ,count(i_current_price),count(i_brand_id)
      ,count(i_brand),count(i_class_id)
      ,count(i_class),count(i_category_id)
      ,count(i_category),count(i_manufact_id)
      ,count(i_manufact),count(i_size)
      ,count(i_formulation),count(i_color)
```

```

        ,count(i_units),count(i_container)
        ,count(i_manager_id),count(i_product_name)
from store_sales join item on ss_item_sk = i_item_sk
where i_size = 'N/A'

```

Before Txn1 completes, submit an ACID Transaction Txn2.

If Txn2 completes before Txn1 completes, run the ACID Query and write the results to file2 and compare file1 and file2 to verify that the appropriate rows in the ITEM table have been changed. In this case, the test is complete with only Steps 1 and 2. If Txn2 will not complete before Txn1 completes, perform Steps 3 and 4:

3. Ensure that Txn1 is still active. Submit a third transaction Txn3, which executes ACIDQ1 (with i\_size <> 'N/A') against the qualification database with a test-sponsor selected value of the substitution parameter [delta] that is not equal to the one used in Step 1.
4. Verify that Txn2 completes before Txn3, and that the appropriate rows in the ITEM table have been changed.

**Comment:** In some implementations Txn2 will not queue behind Txn1. If Txn2 completes prior to Txn1 completion, it is not necessary to run Txn3 in order to demonstrate that updates will be processed in a timely manner as required by Clause 6.4.2.

## 6.5 Durability Requirements

### 6.5.1.1 General Requirements

The SUT must demonstrate durability: the ability to preserve the effects of committed transactions and ensure database consistency after recovery from any one of the failures listed in Clause 6.5.4.

**Comment:** No system provides complete durability (i.e., durability under all possible types of failures). For the purposes of TPC-DS, these tests are deemed sufficient for the single failures specified in Clause 6.5.4.

Since database objects can be stored on devices with more than one protection mechanism (e.g. unprotected, RAID-5, RAID-1, remote replication), the sponsor must demonstrate durability for each mechanism used. For example, if the implementation uses RAID-1 for dimension tables and RAID-5 for fact tables and unprotected storage for auxiliary structures, then all of the durability tests must be run three times. One test for each of the three protection mechanisms must be run.

### 6.5.2 Durable Medium Definition

A durable medium is a data storage medium that is either:

- a) An inherently non-volatile medium (e.g., magnetic disk, magnetic tape, optical disk, solid state disk, etc.) or;
- b) A volatile medium with its own self-contained power supply that will retain and permit the transfer of data, before any data is lost, to an inherently non-volatile medium after the failure of external power.

**Comment:** A configured and priced Uninterruptible Power Supply (UPS) is not considered external power.

**Comment:** A durable medium can fail; this is usually protected against by replication on a second durable medium (e.g., mirroring) or logging to another durable medium. Memory can be considered a durable medium if it can preserve data long enough to satisfy the requirement (b) above. For example, if memory is accompanied by an Uninterruptible Power Supply, and the contents of memory can be transferred to an inherently non-volatile medium during the failure, then the memory is considered durable. Note that no distinction is made between main memory and memory performing similar permanent or temporary data storage in other parts of the system (e.g., disk controller caches).

**Comment:**

### 6.5.3 Committed Property Definition

6.5.3.1 A transaction is considered committed when the transaction manager component of the system has either written the log or written the data for the committed updates associated with the transaction to a durable medium.

**Comment:** Transactions can be committed without the user subsequently receiving notification of that fact, since message integrity is not required for TPC-DS.

**Comment:** Although the order of operations in the ACID Transaction is immaterial, the actual return of data cannot begin until the commit operation has successfully completed.

6.5.3.2 To facilitate the execution of the durability tests, the driver must maintain a durable success file that records the details of each transaction which has successfully completed and whose message has been returned to the driver. At the time of an induced failure, this success file must contain a record of all transactions which have been committed, except for transactions whose commit notification message to the driver was interrupted by the failure.

The durability success file is required only for the durability tests and must contain the following fields:

Table 6-3 Mandatory fields in the durability success file

Fields	Datatype	Definition
I_KEY	identifier	Foreign reference to I_ITEM_ID
DATE_T	date	date and time to second

**Comment:** If the driver resides on the SUT, the success file must be isolated from the TPC-DS database. For example, the success file must be written outside of the ACID Transaction. If the durability of the success file is provided by the same data manager as the TPC-DS database, it must use a different log file.

#### 6.5.4 Single Failures Definitions

The test sponsor shall guarantee the test system will preserve the database and the effects of committed updates after recovery from any of the failures listed below:

Permanent irrecoverable failure of any single durable medium containing TPC-DS database tables, EADS, or recovery log data. The medium to be failed is to be chosen at random by the auditor, and cannot be specially prepared

**Comment:** If main memory is used as a durable medium, then it shall be considered as a potential single point of failure. Sample mechanisms to survive single durable medium failures are database archiving in conjunction with a redo (after image) log, and mirrored durable media. If memory is the durable medium and mirroring is the mechanism used to ensure durability, then the mirrored memories shall be independently powered.

Instantaneous interruption (system crash/system hang) in processing that requires a system re-boot to recover

**Comment:** Instantaneous interruption implies abnormal system shutdown that requires loading of a fresh copy of the operating system from the boot device. It does not necessarily imply loss of volatile memory. When the recovery mechanism relies on the pre-failure contents of volatile memory, the means used to avoid the loss of volatile memory (e.g., an Uninterruptible Power Supply) must be included in the system cost calculation. A sample mechanism to survive an instantaneous interruption in processing is an undo/redo log.

- Failure of all or part of memory (loss of contents)

**Comment:** Memory failure implies that all or part of memory has failed. This failure may be caused by a loss of external power or the permanent failure of a memory board.

- SUT Power Failure: Loss of all external power to the SUT for an indefinite time period.

**Comment:** To demonstrate cluster durability during a power failure, the largest subset of the SUT maintained by a single UPS must be failed. For example, if a system has one UPS per node or set of nodes, it is sufficient to fail one node or that set of nodes. If there is only one UPS for the entire system, then the entire system must be failed. In either case, all UPSs must be priced.

Regardless of UPS configuration, at least one node of each subset of the nodes in the cluster providing a distinct function must be failed.

#### 6.5.5 Durability Tests

The intent of these tests is to demonstrate that all transactions whose output messages have been received by the driver have in fact been committed in spite of any single failure from the list in Clause 6.5.4 and that all consistency conditions are still met after the database is recovered.

For each of the failure types defined in Clause 6.5.4 perform the following steps:

1. Verify that the ITEM table is initially consistent as defined in Clause 6.3.2.1.
2. Asynchronously submit ACID Transactions from each of at least the number of the execution streams (# query streams + 1 refresh stream) used in the Query Runs. It shall be demonstrated that transactions are in progress at the time of the failure.
3. Wait until at least 10 of the ACID transactions from each stream submitted in Step 2 have completed. Cause the failure selected from the list in Clause 6.5.4. At the time of the failure, it shall be demonstrated that:
  - a) At least one transaction is in flight
  - b) No stream has completed submission of the required minimum number of ACID transactions as defined in Step 2

**Comment:** The intent is that the failure is induced while all streams are continuously submitting and executing transactions. If the number of in-flight transactions at the point of failure is less than the number of streams, this is assumed to be a random consequence of interrupting some streams during the very small interval between committing one transaction and submitting the next.

4. Restart the system under test using normal recovery procedures.
5. Compare the contents of the durability success file and the ITEM table to verify that records in the success file for a committed ACID Transaction have a corresponding record in the ITEM table and that no success record exists for uncommitted transactions. Count the number of entries in the success file and the number of new records in the ITEM table and report any difference.

**Comment:** This difference can only be due to transactions that were committed on the system under test, but for which the data was not written in the success file before the failure.

6. Re-verify the consistency of the ITEM tables as defined in Clause 6.3.2.

## 7 Performance Metrics and Execution Rules

### 7.1 Definition of Terms

- 7.1.1 The **Benchmark** is defined as the execution of the **Load Test** followed by the **Performance Test**.
- 7.1.2 The **Load Test** is defined as all activity required to bring the **System Under Test** to the configuration that immediately precedes the beginning of the **Performance Test**. The **Load Test** may not include the execution of any of the queries in the **Power Test** or **Throughput Test** or any similar query.
- 7.1.3 The **Performance Test** is defined as the **Power Test** and both **Throughput Tests**.
- 7.1.4 A **session** is defined as a uniquely identified process context capable of supporting the execution of user-initiated database activity. A **query session** is a session executing activity on behalf of a Query Run. A **refresh session** is one executing activity on behalf of a refresh set.
- 7.1.5 A **query stream** is defined as the sequential execution of a permutation of queries submitted by a single emulated user. A query stream consists of the 99 queries defined in Clause 4.
- 7.1.6 A **refresh set** is defined as the execution of one set of data maintenance functions. A refresh set is implemented in one or more refresh sessions.
- 7.1.7 A **Query Run** consists of  $S_q$  query sessions each running a query stream.
- 7.1.8 A **Refresh Group** is the activity performed on the SUT to execute the refresh sets required for a Query Run.
- 7.1.9 A **Power Test** consists of exactly one query session running a single query stream.
- 7.1.10 A **Throughput Test** consists of one Query Run and one Refresh Group.
- 7.1.11 A **query** is an ordered set of one or more valid SQL statements resulting from applying the required parameter substitutions to a given query template. The order of the SQL statements is defined in the query template.
- 7.1.12 The **SUT** consists of a collection of configured components used to complete the benchmark.
- 7.1.13 The mechanism used to submit queries to the SUT and to measure their execution time is called a **driver**.
- 7.1.14 A **timestamp** must be taken in the time zone the SUT is located in. It is defined as any representation equivalent to yyyy-mm-dd hh:mm:ss.s, where:
  - yyyy is the 4 digit representation of year
  - mm is the 2 digit representation of month
  - dd is the 2 digit representation of day
  - hh is the 2 digit representation of hour in 24-hour clock notation
  - mm is the 2 digit representation of minute
  - ss.s is the 3 digit representation of second with a precision of at least 1/10 of a second
- 7.1.15 **Elapsed time** is measured in seconds rounded up to the nearest 0.1 second.

### 7.2 Configuration Rules

- 7.2.1 The driver is a logical entity that can be implemented using one or more physical programs, processes, or systems (see Clause 8.3).
- 7.2.2 The communication between the driver and the SUT must be limited to one session per query. These sessions are prohibited from communicating with one another except for the purpose of scheduling Data Maintenance functions (see Clause 5.3).

- 7.2.3 All query sessions must be initialized in exactly the same way. All refresh sessions must be initialized in exactly the same way. The initialization of a refresh session may be different than that of the query session.
- 7.2.4 All session initialization parameters, settings and commands must be disclosed.
- Comment:** The intent of this clause is to provide the information needed to precisely recreate the execution environment of any given stream as it exists prior to the submission of the first query or data maintenance function.
- 7.2.5 The attributes of the query sessions shall be the same as the attributes of the session used by the ACID Query (see Clause 6.1.5). Similarly, the attributes of the refresh session(s) shall be the same as the attributes of the session used by the ACID Transaction.
- 7.2.6 The driver shall submit each TPC-DS query for execution by the SUT via the session associated with the corresponding query stream.
- 7.2.7 In the case of the data maintenance functions, the driver is only required to submit the commands necessary to cause the execution of each data maintenance function.
- 7.2.8 The driver's submittal of the queries to the SUT during the performance test shall be limited to the transmission of the query text to the DBMS and whatever additional information is required to conform to the measurement and data gathering requirements defined in this document. In addition:
- The interaction between the driver and the SUT shall not have the purpose of indicating to the SUT or any of its components an execution strategy or priority that is time-dependent or query-specific;
  - The interaction between the driver and the SUT shall not have the purpose of indicating to the SUT, or to any of its components, the insertion of time delays;
  - The driver shall not insert time delays before, after, or between the submission of queries to the SUT;
  - The interaction between the driver and the SUT shall not have the purpose of modifying the behavior or configuration of the SUT (i.e., DBMS or operating system settings) on a query-by-query basis. These parameters shall not be altered during the execution of the performance test.
- Comment:** One intent of this clause is to prohibit the pacing of query submission by the driver.
- 7.2.9 Environmental Assumptions
- 7.2.9.1 The configuration and initialization of the SUT, the database, or the session, including any relevant parameter, switch or option settings, shall be based only on externally documented capabilities of the system that can be reasonably interpreted as useful for a decision support workload. This workload is characterized by:
- Sequential scans of large amounts of data;
  - Aggregation of large amounts of data;
  - Multi-table joins;
  - Possibly extensive sorting.
- 7.2.9.2 While the configuration and initialization can reflect the general nature of this expected workload, it shall not take special advantage of the limited functions actually exercised by the benchmark. The queries actually chosen in the benchmark are merely examples of the types of queries that might be used in such an environment, not necessarily actual user queries. Due to this limit in the scope of the queries and test environment, TPC-DS has chosen to restrict the use of some database technologies (see Clause 2.5). In general, the effect of the configuration on benchmark performance should be representative of its expected effect on the performance of the class of applications modeled by the benchmark.
- 7.2.9.3 The features, switches or parameter settings that comprise the configuration of the operating system, the DBMS or the session must be such that it would be reasonable to expect a database administrator with the following characteristics be able to decide to use them:
- Knowledge of the general characteristics of the workload as defined above;
  - Knowledge of the logical and physical database layout;
  - Access to operating system and database documentation;

- No knowledge of product internals beyond what is documented externally.

Each feature, switch or parameter setting used in the configuration and initialization of the operating system, the DBMS or the session must meet the following criteria:

- It shall remain in effect without change throughout the performance test;
- It shall not make reference to specific tables, indices or queries for the purpose of providing hints to the query optimizer.

7.2.10

The collection of statistics requested through the use of directives must be part of the database load. If these directives request the collection of different levels of statistics for different columns, they must adhere to the following rules.:

- 1) The level of statistics collected for a given column must be based on the column's membership in a class.
- 2) Class definitions must rely solely on the following column attributes from the logical database design (as defined in Clause 2):
  - Datatype;
  - Nullable;
  - Foreign Key;
  - Primary Key.
- 3) Class definitions may combine column attributes using AND, OR and NOT operators. (for example, one class could contain all columns satisfying the following combination of attributes: [Identifier Datatype] AND [NOT nullable OR Foreign Key]);
- 4) Class membership must be applied consistently on all columns across all tables;
- 5) Statistics that operate in sets, such as distribution statistics, should employ a fixed set appropriate to the scale factor used. Knowledge of the cardinality, values or distribution of a non-key column (as specified in Clause 3) must not be used to tailor statistics gathering.

7.2.11

Profile-Directed Optimization

7.2.11.1

Special rules apply to the use of so-called profile-directed optimization (PDO), in which binary executables are reordered or otherwise optimized to best suit the needs of a particular workload. These rules do not apply to the routine use of PDO by a database vendor in the course of building commercially available and supported database products; such use is not restricted. Rather, the rules apply to the use of PDO by a test sponsor to optimize executables of a database product for a particular workload. Such optimization is permissible if all of the following conditions are satisfied:

- The use of PDO or similar procedures by the test sponsor must be disclosed.
- The procedure and any scripts used to perform the optimization must be disclosed.
- The procedure used by the test sponsor could reasonably be used by a customer on a shipped database executable.
- The optimized database executables resulting from the application of the procedure must be supported by the database software vendor.
- The workload used to drive the optimization is described in Clause 7.2.11.2.
- The same set of DBMS executables must be used for all phases of the benchmark.

7.2.11.2

If profile-directed optimization is used, the workload used to drive it can be the execution of any subset of the TPC-DS queries or any data maintenance functions, in any order, against a TPC-DS database of any desired scale factor, with default substitution parameters applied. The query/data maintenance function set, used in PDO, must be reported.

7.3

Query Validation

7.3.1

All query templates used in a benchmark submission shall be validated.

7.3.2

The validation process is defined as follows:



1. Populate the qualification database (see Clause 0) ;
2. Execute the query template using qualification substitution parameters as defined in 11.9.1.1Appendix B;;
3. Compare the output to the answer set defined for the query.

7.3.3 The output data shall match the answer set defined for the query, subject to the constraints defined in Clause 7.5

#### 7.4 Execution Rules

##### 7.4.1 General Requirements

7.4.1.1 If the load test, power test or either throughput test fails, the benchmark run is invalid.

7.4.1.2 All tables created with explicit directives during the execution of the benchmark tests must meet the ACID properties defined in Clause 6.

7.4.1.3 The SUT, including any database server(s), shall not be restarted at any time after the power test begins until after all tests have completed.

7.4.1.4 The driver shall submit queries through one or more sessions on the SUT. Each session corresponds to one, and only one, query stream on the SUT.

7.4.1.5 Parallel activity within the SUT directed toward the execution of a single query or data maintenance function (e.g. intra-query parallelism) is not restricted.

7.4.1.6 The real-time clock used by the driver to compute the timing intervals must measure time with a resolution of at least 0.01 second.

7.4.2 The benchmark must use the following sequence of tests:

- c) Database Load Test
- d) Power Test
- e) Throughput Test 1
- f) Throughput Test 2

##### 7.4.3 Database Load Test

7.4.3.1 The process of building the test database is known as database load. Database load consists of timed and un-timed components.

7.4.3.2 The population of the test database, as defined in Clause 2.1, consists of two logical phases:

- a) Generation: the process of using dsdgen to create data in a format suitable for presentation to the DBMS load facility. The generated data may be stored in memory, or in flat files on tape or disk.
- b) Loading: the process of storing the generated data into the database tables.
- c) Generation and loading of the data can be accomplished in two ways:
- d) **Load from flat files:** dsdgen is used to generate flat files that are stored on disk or tape. The records in these files may optionally be permuted and relocated to the SUT. After table creation on the SUT, data is loaded from the flat files into the database. In this case, only the loading phase contributes to the database load time.
- e) **In-line load:** dsdgen is used to generate data that is directly loaded into the database tables using an "in-line" load facility. In this case, generation and loading occur concurrently and both contribute to the database load time.

7.4.3.3 The resources used to generate, permute, relocate to the SUT or hold dsdgen data may optionally be distinct from those used to run the actual benchmark. For example:

- a) For load from flat files, a separate system or a distinct storage subsystem may be used to generate, store and permute dsdgen data into the flat files used for the database load.
- b) For in-line load, separate and distinct processing elements may be used to generate and permute data and to deliver it to the DBMS.

7.4.3.4 Resources used only in the generation phase of the population of the test database must be treated as follows:

For load from flat files,

- a) Any processing element (e.g., CPU or memory) used exclusively to generate and hold dsdgen data or relocate it to the SUT prior to the load phase shall not be included in the total priced system and shall be physically removed from or made inaccessible to the SUT prior to the start of the load phase using vendor supported method;
- b) Any storage facility (e.g., disk drive, tape drive or peripheral controller) used exclusively to generate and deliver data to the SUT during the load phase shall not be included in the total priced system. The test sponsor must demonstrate to the satisfaction of the auditor that this facility is not being used in the performance tests.

For in-line load, any processing element (e.g., CPU or memory) or storage facility (e.g., disk drive, tape drive or peripheral controller) used exclusively to generate and deliver dsdgen data to the SUT during the load phase shall not be included in the total priced system and shall be physically removed from or made inaccessible to the SUT prior to the start of the measurement tests.

**Comment:** The intent is to isolate the cost of resources required to generate data from those required to load data into the database tables.

7.4.3.5 An implementation may require additional programs to transfer dsdgen data into the database tables (from either flat file or in-line load). If non-commercial programs are used for this purpose, their source code must be disclosed. If commercially available programs are used for this purpose, their vendors and configurations shall be disclosed. Whether or not the software is commercially available, use of the software's functionality's shall be limited to:

1. Permutation of the data generated by dsdgen ;
2. Delivery of the data generated by dsdgen to the DBMS.

7.4.3.6 The database load must be implemented using commercially available utilities (invoked at the command level or through an API) or an SQL programming interface (such as embedded SQL or ODBC).

7.4.3.7 Database Load Time

7.4.3.7.1 The elapsed time to prepare the test database for the execution of the performance test is called the Database Load Time ( $T_{LOAD}$ ), and must be disclosed. It includes all of the elapsed time to create the tables defined in Clause 2.1, load data, create and populate EADS, define and validate constraints, gather statistics for the test database, configure the system under test to execute the performance test, and to ensure that the test database meets the ACID requirements including syncing loaded data on RAID devices and the taking of a backup of the database, when necessary.

7.4.3.8 The Database Load Time, known as  $T_{LOAD}$  is the difference between Load Start Time and Load End Time.

- Load Start Time is defined as the timestamp taken at the start of the creation of the tables defined in Clause 2.1 or when the first character is read from any of the flat files or, in case of in-line load, when the first character is generated by dsdgen, whichever happens first
- Load End Time is defined as the timestamp taken when the database is fully populated, all EADS are created, a database backup has completed (if applicable) and the SUT is configured, as it will be during the performance test.

**Comment:** Since the time of the end of the database load is used to seed the random number generator for the substitution parameters, that time cannot be delayed in any way that would make it predictable to the test sponsor.

7.4.3.8.1 There are five classes of operations which may be excluded from database load time:

- a) Any operation that does not affect the state of the DBMS (e.g., data generation into flat files, relocation of flat files to the SUT, permutation of data in flat files, operating-system-level disk partitioning or configuration);
- b) Any modification to the state of the DBMS that is not specific to the TPC-DS workload (e.g. logical tablespace creation or database block formatting);

- c) The time required to install or remove physical resources (e.g. CPU, memory or disk) on the SUT that are not priced;
- d) An optional backup of the test database performed at the test sponsor's discretion. However, if a backup is required to ensure that the ACID properties can be met, it must be included in the load time;
- e) Operations that create RAID devices.
- f) Tests required to fulfill data validation test (see Clause 3.5)

7.4.3.8.2 There cannot be any manual intervention during the Database Load.

7.4.3.8.3 The SUT or any component of it must not be restarted after the start of the load test and before the start of the performance test.

**Comment:** The intent of this Clause is that when the timing ends the system under test be capable of executing the performance test without any further change. The database load may be decomposed into several phases. Database load time is the sum of the elapsed times of all phases during which activity other than that detailed in Clause 7.4.3.8.1 occurred on the SUT.

7.4.4 Power Test

7.4.4.1 The Power Test is executed immediately following the load test.

7.4.4.2 The Power Test measures the ability of the system to process a sequence of queries in the least amount of time in a single stream fashion.

7.4.4.3 For each query in the Power Test, at least one atomic transaction shall be started and completed.

7.4.4.4 The Power Test shall execute queries submitted by the driver through a single query stream with stream identification number 0 and using a single session on the SUT.

7.4.4.5 The queries in the Power Test shall be executed in the order assigned to its stream identification number and defined in 11.9.1.1Appendix D:.

7.4.4.6 Only one query shall be active at any point of time during the Power Test.

7.4.4.7 The elapsed time of the Power Test ( $T_{\text{Power}}$ , as defined in Clause 7.4.8.3) must not exceed 6 hours.

7.4.5 Throughput Tests

7.4.5.1.1 The Throughput Tests measure the ability of the system to process the most queries in the least amount of time with multiple users and data maintenance activity.

7.4.5.1.2 Throughput Test 1 immediately follows the Power Test. Throughput Test 2 follows the successful completion of Throughput Test 1.

7.4.5.1.3 Throughput Test 1 consists of Query Run 1 and Refresh Group 1. Throughput Test 2 consists of Query Run 2 and Refresh Group 2.

7.4.5.1.4 Any explicitly created aggregates, as defined in Clause 5.1.5, present and enabled during any portion of Query Run 1 or 2 must be present and enabled at all times that queries are being processed.

7.4.5.1.5 Each query stream contains a distinct permutation of the query templates defined for TPC-DS. The permutation of queries for the first 20 query streams is shown in 11.9.1.1Appendix D:.

7.4.6 Throughput Test Timing

7.4.6.1 The elapsed time of Throughput Test 1, known as  $T_{\text{TT1}}$  is the difference between Throughput Test 1 Start Time and Throughput Test 1 End Time,

- 7.4.6.2 Throughput Test 1 Start Time is defined as a timestamp identical to Query Run 1 Start Time.
- Query Run 1 Start Time, which is the timestamp that must be taken before the first character of the executable query text of the first query of the first query stream is submitted to the SUT by the driver.
- 7.4.6.3 Throughput Test 1 End Time is defined as the later of:
- Query Run 1 End Time, which is the timestamp that must be taken after the last character of output data from the last query of the last query stream is received by the driver from the SUT or
  - Refresh Group 1 End Time, which is the ending timestamp of the last refresh set in Throughput Test 1, including all EADS updates.
- Comment:** The intent of Clauses 7.4.6.1 through 7.4.6.3 is that the Throughput Test 1 elapsed time will be from the start of Query Run 1 until the end of either Query Run 1 or Refresh Group 1, whichever ends last.
- Comment:** In this clause a query stream is said to be first if it starts submitting queries before any other query streams. The last query stream is defined to be that query stream whose output data are received last by the driver.
- 7.4.6.4 The elapsed time of Throughput Test 2, known as  $T_{TT2}$  is the difference between Throughput Test 2 Start Time and Throughput Test 2 End Time,
- 7.4.6.5 Throughput Test 2 Start Time is defined as a timestamp identical to Throughput Test 1 End Time.
- 7.4.6.6 Throughput Test 2 End Time is defined as the later of:
- Query Run 2 End Time, which is the timestamp that must be taken after the last character of output data from the last query of the last query stream is received by the driver from the SUT or
  - Refresh Group 2 End Time, which is the ending timestamp including all EADS updates.
- Comment:** The intent of Clauses 7.4.6.4 through 7.4.6.6 is that the Throughput Test 2 elapsed time will be from the end of Throughput Test 1 until the end of either Query Run 2 or Refresh Group 2, whichever ends last.
- 7.4.7 Query Run
- 7.4.7.1 Only one query shall be active on any of the sessions at any point of time during a Query Run.
- 7.4.7.2 For each query, at least one atomic transaction shall be started and completed.
- 7.4.7.3 The Query Run shall execute queries submitted by the driver through a sponsor-selected number of query streams ( $S_q$ ). There must be one session per query stream on the SUT and each stream must execute queries serially (i.e. one after another).
- 7.4.7.4 Each query stream is uniquely identified by a **stream identification number**  $s$  ranging from 1 to  $S$ , where  $S$  is the number of query streams in the Throughput Tests (Query Run 1 plus Query Run 2).
- 7.4.7.5 Once a stream identification number has been generated and assigned to a given query stream, the same number must be used for that query stream for the duration of the test.
- 7.4.7.6 The value of  $S_q$  is any even number larger than or equal to 4.
- 7.4.7.7 The same value of  $S_q$  shall be used for both Query Runs, and shall remain constant throughout each Query Run.
- 7.4.7.8 Queries must be interspersed with refresh sets. The requirements for scheduling queries and refresh sets are defined in clauses 7.4.9.5 and 7.4.9.6. This mechanism must be implemented by the driver.
- 7.4.7.9 The queries in each query stream shall be executed in the order assigned to the stream identification number and defined in 11.9.1.1 Appendix D:.
- 7.4.8 Query Timing
- 7.4.8.1 For a given query template  $t$ , used to produce the  $i^{\text{th}}$  query within query stream  $s$ , the **query elapsed time**,  $QD(s, i, t)$ , is the difference between:
- The timestamp when the first character of the executable query text is submitted to the SUT by the driver;

- The timestamp when the last character of the output is returned from the SUT to the driver and a success message is sent to the driver.

**Comment:** All the operations that are part of the execution of a query (e.g., creation and deletion of a temporary table or a view) must be included in the elapsed time of that query.

7.4.8.2 The elapsed time of each query in each stream shall be disclosed for each Query Run and Power Test.

7.4.8.3 The elapsed time of the Power Test, known as  $T_{\text{Power}}$  is the difference between

- Power Test Start Time, which is the timestamp that must be taken before the first character of the executable query text of the first query of Stream 0 is submitted to the SUT by the driver; and
- Power Test End Time, which is the timestamp that must be taken after the last character of output data from the last query of Stream 0 is received by the driver from the SUT.

7.4.8.4 The elapsed time of Query Run 1, known as  $T_{\text{QR1}}$  is the difference between:

- Query Run 1 Start Time, which is the timestamp that must be taken before the first character of the executable query text of the first query of the first query stream is submitted to the SUT by the driver; and

**Comment:** In this clause a query stream is said to be first if it starts submitting queries before any other query streams.

- Query Run 1 End Time, which is the timestamp that must be taken after the last character of output data from the last query of the last query stream is received by the driver from the SUT.

**Comment:** In this clause the last query stream is defined to be that query stream whose output data are received last by the driver.

7.4.8.5 The elapsed time of Query Run 2, known as  $T_{\text{QR2}}$  is the difference between:

- Query Run 2 Start Time, which is the timestamp that must be taken before the first character of the executable query text of the first query of the first query stream of Query Run 2 is submitted to the SUT by the driver; and
- Query Run 2 End Time, which is the timestamp that must be taken after the last character of output data from the last query of the last query stream is received by the driver from the SUT.

**Comment:** In this clause the last query stream is defined to be that query stream whose output data are received last by the driver.

7.4.8.6 The elapsed time of each Query Run and Power Test shall be disclosed.

7.4.9 Refresh Groups

7.4.9.1 The Refresh Groups measure the ability to perform desired data changes interspersed among queries.

7.4.9.2 Each refresh group shall execute  $S_q/2$  refresh sets.

7.4.9.3 Each refresh set uses its own data set as generated by dsdgen. Refresh sets must be executed in the order generated by dsdgen.

7.4.9.4 Any explicitly created aggregates, as defined in clause 5.1.5, present and enabled during any portion of Query Run 1 must conform to clause 7.4.5.1.4.

7.4.9.5 Data maintenance operations may execute concurrently with queries or alone. Refresh sets do not overlap; at most one refresh set is running at any time. All data maintenance functions need to have finished on refresh data set  $n$  before any data maintenance function can commence on refresh data set  $n+1$ .

**Comment:** Each set of data maintenance functions runs with its own refresh data set. The order of refresh data sets is determined by dsdgen.

7.4.9.6 During the throughput run the following two requirements define the sequencing between queries and refresh sets:

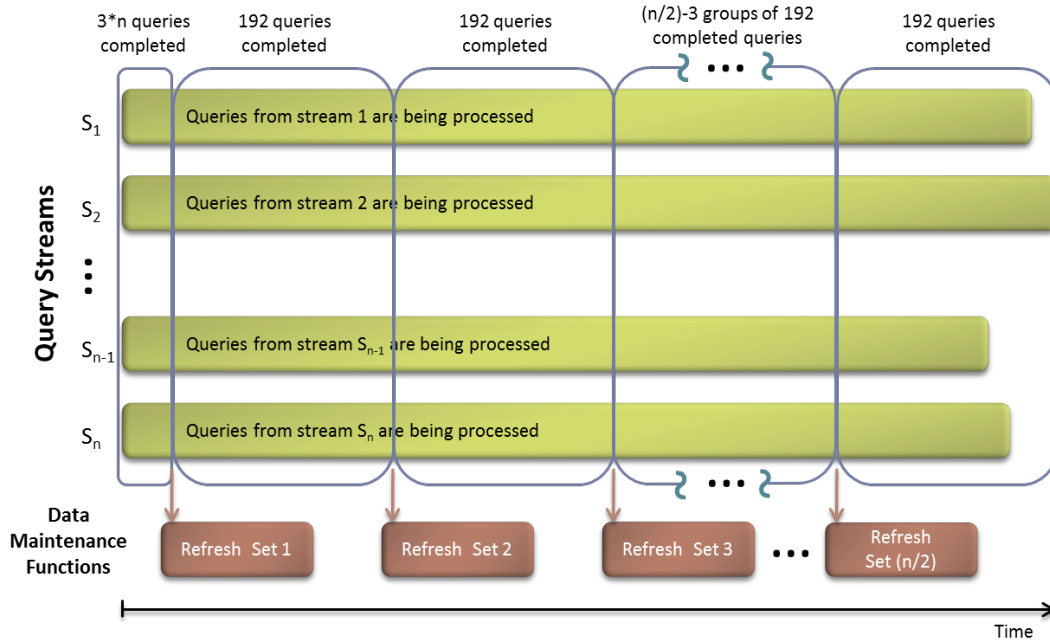
- The  $N^{\text{th}}$  refresh set can only start after  $[(3*S_q) + ((N-1)*192)]$  queries have completed (aggregated over all streams).

- The  $[(3*S_q) + (N*192) + 1]^{th}$  query (aggregated over all streams) can only start after the  $N^{th}$  refresh set has completed.

For example:

- At least  $(3*S_q)$  queries must complete before the first refresh set can start.
- At least 192 additional queries must complete before the second refresh set can start.
- At least  $(3*S_q) + ((N-1)*192)$  queries must complete before the  $N^{th}$  refresh set can start.
- The first refresh set must complete before starting more than  $(3*S_q) + 192$  queries.
- The  $N^{th}$  refresh set must complete before starting more than  $(3*S_q) + (N*192)$  queries.

**Comment:** The purpose of linking data maintenance operations to completion of queries is so that the updates are interspersed among execution of queries in the throughput runs, although concurrent execution of updates and queries is not required. The figure below shows an example of how the data maintenance functions can be sequenced in relation to the completion of queries across all query streams.



7.4.9.7 The scheduling of each data maintenance function within Refresh Sets is left to the test sponsor.

7.4.10 Refresh Timing

7.4.10.1 The elapsed time,  $DI(i,s)$ , for the execution of the data maintenance function  $i$ , of the  $s^{th}$  refresh set (e.g. applying the  $s^{th}$  refresh data set on the data maintenance function  $i$ ), is the difference between:

- The timestamp,  $DS(i,s)$ , when the first character of the data maintenance function  $i$  executing on refresh data set  $s$  is submitted to the SUT by the driver, or when the first character requesting the execution of Data Maintenance function  $i$  is submitted to the SUT by the driver, whichever happens first;
- The timestamp,  $DE(i,s)$  end time, when the last character of output data from the last data maintenance function of the last refresh data set is received by the driver from the SUT and a success message has been received by the driver from the SUT.

7.4.10.2 The elapsed time,  $DI(s)$ , for the execution of all data maintenance functions of refresh set  $s$  is the difference between the start timestamp of refresh set  $s$ ,  $DS(s)$  and the end timestamp of refresh set  $s$ ,  $DE(s)$ .  $DS(s)$  is defined as  $DS(i,s)$ , where  $i$  denotes the first data maintenance function executed in refresh set  $s$ .  $DE(s)$  is defined as  $DE(j,s)$ , where  $j$  is the last data maintenance function executed in refresh set  $s$ .

7.4.10.3 The elapsed time of each data maintenance function within each refresh set must be disclosed, i.e. all  $DI(i,s)$  must be disclosed.

7.4.10.4 The timestamp of the start and end times and the elapsed time of each refresh set must be disclosed, i.e. for all refresh sets s DS(s), DE(s) and DI(s) must be disclosed.

## 7.5 Output Data

7.5.1 After execution, a query returns one or more rows. The rows are called the output data.

7.5.2 Output data shall adhere to the following guidelines:

- a) Columns appear in the order specified by the SELECT list of the query.
- b) Column headings are optional.
- c) Non-integer expressions including prices are expressed in decimal notation with at least two digits behind the decimal point.
- d) Integer quantities contain no leading zeros.
- e) Dates are expressed in a format that includes the year, month and day in integer form, in that order (e.g., YYYY-MM-DD). The delimiter between the year, month and day is not specified. Other date representations, for example the number of days since 1970-01-01, are specifically not allowed.
- f) Strings are case-sensitive and must be displayed as such. Leading or trailing blanks are acceptable.
- g) The amount of white space between columns is not specified.
- h) The order of a query output data must match the order of the validation output data, except for queries that do not specify an order for their output data.

**Comment:** The intent of this clause is to assure that output data is expressed in a format easily readable by a non-sophisticated computer user, and can be compared with known output data for query validation.

7.5.3 The precision of all values contained in the output data shall adhere to the following rules:

- a) For singleton column values and results from COUNT aggregates, the values must exactly match the query validation output data.
- b) For ratios, results must be within 1% of the query validation output data when reported to the nearest 1/100th, rounded up.
- c) For results from SUM money aggregates, the resulting values must be within \$100 of the query validation output data.
- d) For results from AVG aggregates, the resulting values must be within 1% of the query validation output data when reported to the nearest 1/100th, rounded up.

## 7.6 Metrics

7.6.1 TPC-DS defines three primary metrics:

- a) A Performance Metric, QphDS@SF, reflecting the TPC-DS query throughput (see Clause 7.6.3);
- b) A Price-Performance metric, \$/QphDS@SF (see Clause 7.6.4);
- c) System availability date (see Clause 7.6.5).

7.6.2 TPC-DS also defines several secondary metrics. The secondary metrics are:

- a) Load time, as defined in Clause 7.4.3.7;
- b) Power Test Elapsed time as defined in Clause 7.4.4 and the elapsed time of each query in the Power Test;
- c) Throughput Test 1 and Throughput Test 2 elapsed times, as defined in clause 7.4.6.
- d) When TPC\_Energy option is chosen for reporting, the TPC-DS energy metric reports the power per performance and is expressed as Watts/QphDS@SF. (see TPC-Energy specification for additional requirements).

Each secondary metric shall be referenced in conjunction with the scale factor at which it was achieved. For example, Load Time references shall take the form of Load Time @ SF, or "Load Time = 10 hours @ 300GB".

7.6.3 The Performance Metric (QphDS@SF)

7.6.3.1 The primary performance metric of the benchmark is QphDS@SF, the effective query throughput of the benchmarked configuration, defined as:

$$QphDS@SF = \left\lfloor \frac{SF * Q}{(T_{PT} + T_{TT} + T_{LD})} \right\rfloor$$

Where:

- SF is defined in Clause 3.1.3, and is based on the scale factor used in the benchmark
- Q is the total number of weighted queries:  $Q=3 * S_q * 99$ , with  $S_q$  being the number of streams executed in a Query Run
- $T_{PT}=T_{Power} * S_q$ , where  $T_{PT}$  is the total elapsed time to complete the Power Test, as defined in Clause 7.4.8.3, and  $S_q$  is the number of streams executed in a Query Run
- $T_{TT}=T_{TT1}+T_{TT2}$ , where  $T_{TT1}$  is the total elapsed time of Throughput Test 1 and  $T_{TT2}$  is the total elapsed time of Throughput Test 2, as defined in Clause 7.4.6.
- $T_{LD}$  is the load factor computed as  $T_{LD}=0.01 * S_q * T_{Load}$ , where  $S_q$  is the number of streams executed in a Query Run and  $T_{Load}$  is the time to finish the load, as defined in Clause 7.4.3.7
- $T_{PT}$ ,  $T_{TT}$  and  $T_{LD}$  quantities are in units of decimal hours with a resolution of at least 1/3600<sup>th</sup> of an hour (i.e., 1 second)

**Comment:** The floor symbol ( $\lfloor \rfloor$ ) in the above equation truncates any fractional part.

7.6.4 The Price Performance Metric (\$/QphDS@SF)

7.6.4.1 The price-performance metric for the benchmark is defined as:

$$$/QphDS@SF = \frac{P}{QphDS@SF}$$

Where:

P is the price of the Priced System as defined in Clause 9.1.1.

QphDS@SF is the reported performance metric as defined in Clause 7.6.3

7.6.4.2 If a benchmark configuration is priced in a currency other than US dollars, the units of the price-performance metrics may be adjusted to employ the appropriate currency.

7.6.5 The System Availability Date, as defined in the TPC Pricing Specification Version 1 must be disclosed in any references to either the performance or price-performance metric of the benchmark.

7.6.6 Fair Metric Comparison

7.6.6.1 Results at the different scale factors are not comparable, due to the substantially different computational challenges found at different data volumes. Similarly, the system price/performance may not scale down linearly with a decrease in database size due to configuration changes required by changes in database size.



If results measured against different database sizes (i.e., with different scale factors) appear in a printed or electronic communication, then each reference to a result or metric must clearly indicate the database size against which it was obtained. In particular, all textual references to TPC-DS metrics (performance or price/performance) appearing must be expressed in the form that includes the size of the test database as an integral part of the metric's name; i.e. including the "@size" suffix. This applies to metrics quoted in text or tables as well as those used to annotate charts or graphs. If metrics are presented in graphical form, then the test database size on which metric is based must be immediately discernible either by appropriate axis labeling or data point labeling.

In addition, the results must be accompanied by a disclaimer stating:

"The TPC believes that comparisons of TPC-DS results measured against different database sizes are misleading and discourages such comparisons".

7.6.6.2 Any TPC-DS result is comparable to other TPC-DS results regardless of the number of query streams used during the test (as long as the scale factors chosen for their respective test databases were the same).

#### 7.6.7 Required Reporting Components

To be compliant with the TPC-DS standard and the TPC's fair use policies, all public references to TPC-DS results for a given configuration must include the following components:

- The size of the test database, expressed separately or as part of the metric's names (e.g., QphDS@10GB);
- The TPC-DS Performance Metric, QphDS@Size;
- The TPC-DS Price/Performance metric, \$/QphDS@Size;
- The Availability Date of the complete configuration (see TPC Pricing Specification located on the TPC website (<http://www.tpc.org>)).

Following are two examples of compliant reporting of TPC-DS results:

Example 1: At 10GB the RALF/3000 Server has a TPC-DS Query-per-Hour metric of 3010 when run against a 10GB database yielding a TPC-DS Price/Performance of \$1,202 per query-per-hour and will be available 1-Apr-06.

Example 2: The RALF/3000 Server, which will start shipping on 1-Apr-06, is rated 3,010 QphDS@10GB and 1202 \$/QphDS@10GB.

## 8 SUT AND DRIVER IMPLEMENTATION

This clause defines the System Under Test (SUT) and the benchmark driver.

### 8.1 Models of Tested Configurations

8.1.1 The tested and reported configuration(s) is composed of a driver that submits queries to a system under test (SUT). The SUT executes these queries and replies to the driver. The driver resides on the SUT hardware and software.

8.1.2 Figure 8-1 illustrates examples of driver/SUT configurations. The driver is the shaded area. The diagram also depicts the driver/SUT boundary (see Clause 7.1.15 and Clause 7.4) where timing intervals are measured.

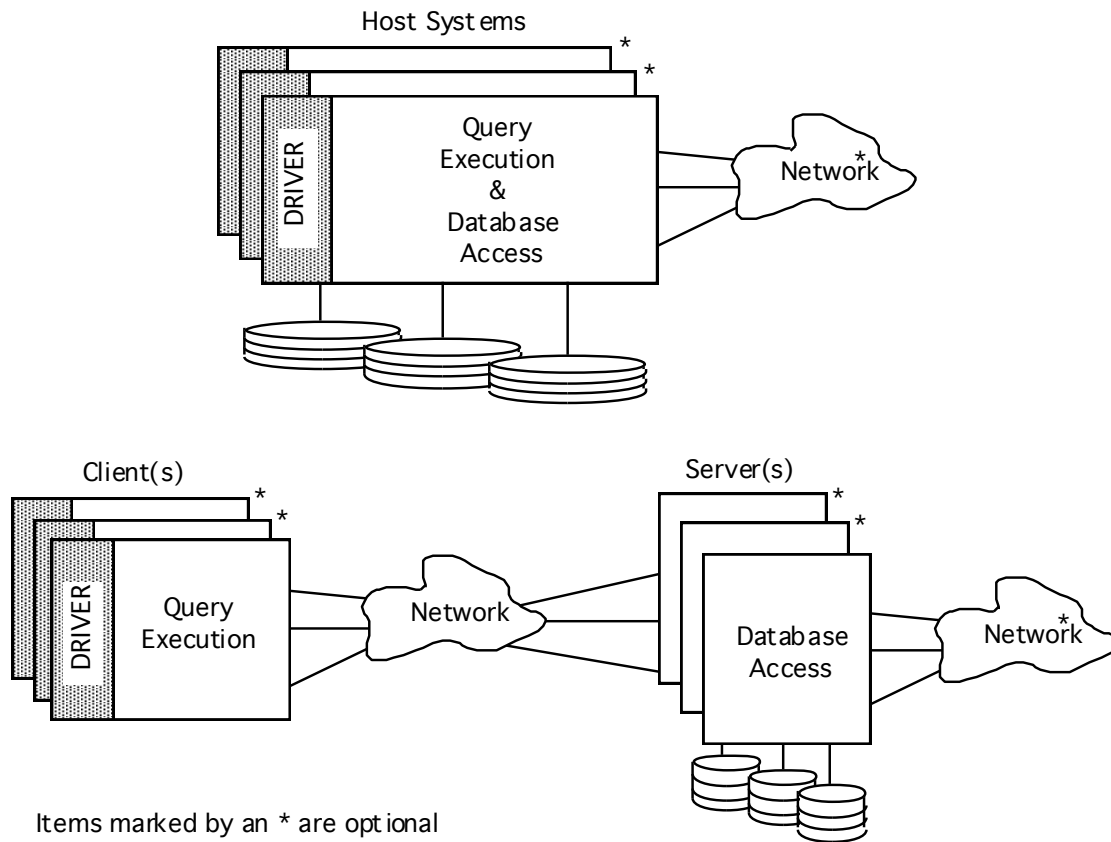


Figure 8-1: Two driver/SUT configurations, a "host-based" and a "client/server" configuration

### 8.2 System Under Test (SUT) Definition

8.2.1 The SUT consists of:

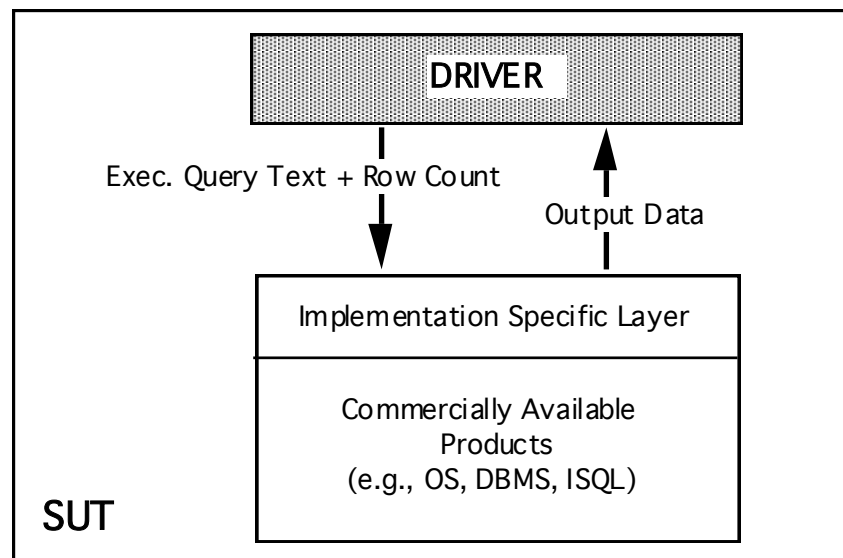
- The host system(s) or server(s), including hardware and software supporting access to the database employed in the performance test and whose cost and performance are described by the benchmark metrics
- Any client processing units (e.g., front-end processors, workstations, etc.) used to execute the queries
- The hardware and software components needed to communicate with user interface devices
- The hardware and software components of all networks required to connect and support the SUT components

- e) Data storage media sufficient to satisfy scaling rules in Clause 3, ACID properties in Clause 6.1 and data described in Clause 7.5.

8.2.2 All SUT components, as described in Clause 8.2.1, shall be commercially available software or hardware products.

8.2.3 An implementation-specific layer can be implemented on the SUT. This layer shall be logically located between the driver and the SUT, as depicted by Figure 8-2.

Figure 8-2: Implementation Specific Layer



8.2.4 If present on the SUT, an implementation-specific layer, , shall be minimal and general purpose (i.e., not limited to the TPC-DS queries). iThe source code shall be disclosed. The functions performed by an implementation specific layer shall be strictly limited to the following:

- a) Database transaction control operations before and after each query execution
- b) Cursor control and manipulation operations around the executable query text
- c) Definition of procedures and data structures required to process dynamic SQL, including the communication of the executable query text to the commercially available layers of the SUT and the reception of the query output data
- d) Communication with the commercially available layers of the SUT
- e) Buffering of the query output data
- f) Communication with the driver

The following are examples of functions that the implementation-specific layer shall not perform:

- a) Any modification of the executable query text;
- b) Any use of stored procedures to execute the queries;
- c) Any sorting or translation of the query output data;
- d) Any function prohibited by the requirements of Clause 7.2.9.1.

### 8.3 Driver Definition

8.3.1 The driver presents the workload to the SUT. The driver is a logical entity that can be implemented using one or more programs, processes, or systems. The driver shall perform only the following functions:

- a) Generate a unique stream ID, starting with 1 for each query stream
- b) Sequence queries for execution by the query
- c) Activate, schedule, and/or synchronize the execution of data maintenance functions
- d) Generate the executable query text for each query
- e) Generate values for the substitution parameters of each query
- f) Complete the executable query text by replacing the substitution parameters by the values generated for them and, if needed, replacing the text-tokens by the query stream ID
- g) Submit each complete executable query text to the SUT for execution, including the number of rows to be returned when specified by the functional query definition
- h) Submit each data maintenance function to the SUT for execution
- i) Receive the output data resulting from each query execution from the SUT
- j) Measure the execution times of the queries and the data maintenance functions and compute measurement statistics
- k) Maintain an audit log of query text and query execution output

8.3.2 The generation of executable query text used by the driver to submit queries to the SUT does not need to occur on the SUT and does not have to be included in any timing interval.

8.3.3 The driver shall not perform any function other than those described in Clause 8.3.1. Specifically, the driver shall not perform any of the following functions:

- a) Performing, activating, or synchronizing any operation other than those mentioned in Clause 8.3.1
- b) Delaying the execution of any query after the execution of the previous query other than for delays necessary to process the functions described in Clause 8.3.1. This delay must be reported and can not exceed half a second between any two consecutive queries of the same query stream
- c) Modifying the compliant executable query text prior to its submission to the SUT
- d) Embedding the executable query text within a stored procedure definition or an application program
- e) Submitting to the SUT the values generated for the substitution parameters of a query other than as part of the executable query text submitted
- f) Submitting to the SUT any data other than the instructions to execute the data maintenance functions, the compliant executable query text and, when specified by the functional query definition, the number of rows to be returned
- g) Artificially extending the execution time of any query.

8.3.4 The driver is not required to be priced.

## 9 PRICING

This section defines the components, functional requirements of what is priced, and what substitutions are allowed. Rules for pricing the Priced Configuration and associated software and maintenance are included in the current revision of the TPC Pricing Specification Version 1 located on the TPC website (<http://www.tpc.org>).

### 9.1 Priced System

The system to be priced shall include the hardware and software components present in the System Under Test (SUT), a communication interface that can support user interface devices, additional operational components configured on the test system, and maintenance on all of the above

#### 9.1.1 System Under Test

Calculation of the priced system consists of:

- Price of the SUT as tested and defined in Clause 8;
- Price of a communication interface capable of supporting the required number of user interface devices defined in Clause 8;
- Price of on-line storage for the database as described in Clause 9.1.3 and storage for all software included in the priced configuration;
- Price of additional products (software or hardware) required for customary operation, administration and maintenance of the SUT for a period of 3 years
- Price of all products required to create, execute, administer, and maintain the executable query texts or necessary to create and populate the test database.

Specifically excluded from the priced system calculation are:

- End-user communication devices and related cables, connectors, and concentrators;
- Equipment and tools used exclusively in the production of the full disclosure report;
- Equipment and tools used exclusively for the execution of the **dsdgen** or **dsqgen** (see Appendix F) programs.

#### 9.1.2 User Interface Devices and Communications

##### 9.1.2.1 The priced system must include the hardware and software components of a communication interface capable of supporting a number of user interface devices (e.g., terminals, workstations, PCs, etc.) at least equal to 10 times the minimum number of query streams or the actual number of query streams, whichever is greater.

**Comment:** Test sponsors are encouraged to configure the SUT with a general-purpose communication interface capable of supporting a large number of user interface devices.

##### 9.1.2.2 Only the interface is to be priced. Not to be included in the priced system are the user interface devices themselves and the cables, connectors and concentrators used to connect the user interface devices to the SUT. For example, in a configuration that includes an Ethernet interface to communicate with PCs, the Ethernet card and supporting software must be priced, but not the Ethernet cables and the PCs.

**Comment:** Active components (e.g., workstations, PCs, concentrators, etc.) can only be excluded from the priced system under the assumption that their role is strictly limited to submitting executable query text and receiving output data and that they do not participate in the query execution. All query processing performed by the tested configuration is considered part of the performance test and can only be done by components that are included in the priced system.

##### 9.1.2.3 The communication interface used must be an industry standard interface, such as Ethernet, Token Ring, or RS232.

9.1.2.4 The following diagram illustrates the boundary between what is priced (on the right) and what is not (on the left). In the event the driver is a commercial product its price should not be included in the price of the SUT:

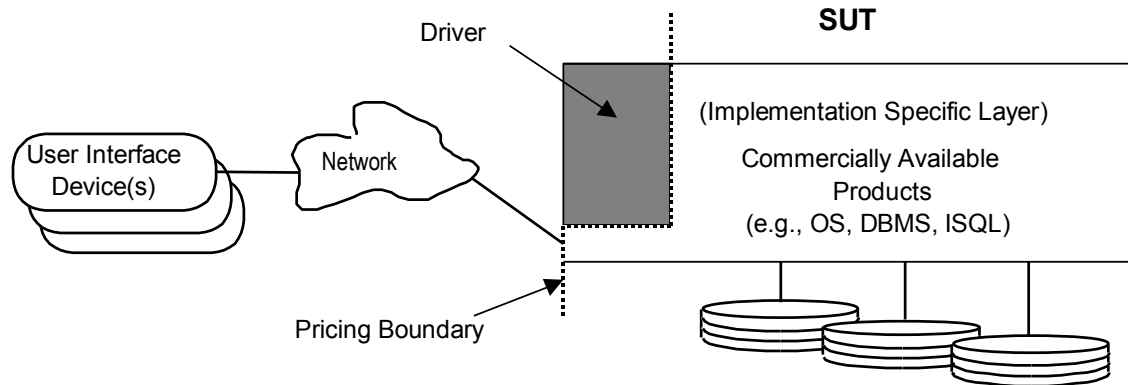


Figure 9-1: Pricing Boundary

### 9.1.3 Database Storage and Recovery Log

9.1.3.1 Recovery data must be maintained for at least the duration of the run used to compute the published performance metrics.(see Clause 7).

9.1.3.2 Roll-back recovery data must be either in memory or in on-line storage at least until all transactions dependent on it are committed. Roll-forward recovery data may be stored on an off-line device provided that:

- The process that stores the roll-forward data is active during the measurement interval;
- The roll-forward data that is stored off-line during the measurement interval must be at least as great as the roll-forward recovery data that is generated during the period (i.e., the data may be first created in on-line storage and then moved to off-line storage, but the creation and the movement of the data must be in steady state);
- All ACID properties must be retained.

**Comment:** Storage is considered on-line if any record can be accessed randomly and updated within 1 second even if this access time requires the creation of a logical access path not present in the tested database. For example, a disk-based sequential file might require the creation of an index to satisfy the access time requirement. On-line storage may include magnetic disks, optical disks, or any combination of these, provided that the above mentioned access criteria are met.

9.1.3.3 While the benchmark requires the configuration of storage sufficient to hold the requisite recovery data as specified in Clause 9.1.3, it does not explicitly require the demonstration of rollforward recovery except as required by the ACID tests (See Clause 5.3.13.19).

9.1.3.4 The storage that is required to be priced includes:

- storage required to execute the benchmark;
- storage to hold recovery data;
- storage and media needed to assure that the test database meets the ACID requirements;
- storage used to hold optional staging area data.

9.1.3.5 All storage required for the priced system must be present on the tested system.

### 9.1.4 Additional Operational Components

9.1.4.1 Additional products that might be included on a customer installed configuration, such as operator consoles and magnetic tape drives, are also to be included in the priced system if explicitly required for the operation, administration, or maintenance, of the priced system.

- 9.1.4.2 Copies of the software, on appropriate media, and a software load device, if required for initial load or maintenance updates, must be included.
- 9.1.4.3 The price of an Uninterruptible Power Supply, if specifically contributing to a durability solution, must be included.
- 9.1.4.4 The price of all cables used to connect components of the system (except as noted in Clause9.1.2.2) must be included.
- 9.2 Allowable Substitution
- 9.2.1 Substitution is defined as a deliberate act to replace components of the Priced Configuration by the Test Sponsor as a result of failing the availability requirements of the TPC Pricing Specification version 1 or when the Part Number for a component changes.
- 9.2.2 Some hardware components of the Priced Configuration may be substituted after the Test Sponsor has demonstrated to the Auditor's satisfaction that the substituting components do not negatively impact the reported TPC-DS Performance Metric. All Substitutions must be reported in the Report and noted in the Auditor's Attestation Letter. The following hardware component may be substituted:
- Durable Medium

## 10 FULL DISCLOSURE

### 10.1 Reporting Requirements

- 10.1.1 A Full Disclosure Report (FDR) is required for a benchmark publication. The FDR is a zip file of a directory structure containing the following:
- 10.1.2 A Report in Adobe Acrobat PDF format,
- 10.1.3 An Executive Summary Statement (ES) in Adobe Acrobat PDF format,
- 10.1.4 An XML document (“ES.xml”) with approximately the same information as in the Executive Summary Statement,
- 10.1.5 The Supporting Files consisting of various source files, scripts, and listing files.
- 10.1.6 Requirements for the FDR file directory structure are described below.
- 10.1.7 The intent of this disclosure is to simplify comparison between results and for a customer to be able to replicate the results of this benchmark given appropriate documentation and products.

### 10.2 Format Guidelines

- 10.2.1 While established practice or practical limitations may cause a particular benchmark disclosure to differ from the examples provided in various small ways, every effort should be made to conform to the format guidelines. The intent is to make it as easy as possible for a reviewer to read, compare and evaluate material in different benchmark disclosures.
- 10.2.2 All sections of the report, including appendices, must be printed using font sizes of a minimum of 8 points.
- 10.2.3 The Executive Summary must be included near the beginning of the full disclosure report.
- 10.2.4 The directory structure of the FDR has three folders:
  - *ExecutiveSummaryStatement* - contains the Executive Summary Statement and ES.xml
  - *Report* - contains the Report,
  - *SupportingFiles* - contains the Supporting Files.

### 10.3 Full Disclosure Report Contents

The FDR should be sufficient to allow an interested reader to evaluate and, if necessary, recreate an implementation of TPC-DS. If any sections in the FDR refer to another section of the report (e.g., an appendix), the names of the referenced scripts/programs must be clearly labeled in each section.

**Comment:** Since the building of a database may consist of a set of scripts and corresponding input files, it is important to disclose and clearly identify, by name, scripts and input files in the FDR.

The order and titles of sections in the test sponsor's full disclosure report must correspond with the order and titles of sections from the TPC-DS standard specification (i.e., this document).

#### 10.3.1 General Items

- 10.3.1.1 A statement identifying the benchmark sponsor(s) and other participating companies must be provided.
- 10.3.1.2 Settings must be provided for all customer-tunable parameters and options that have been changed from the defaults found in actual products, including but not limited to:
  - a) Database tuning options;
  - b) Optimizer/Query execution options;
  - c) Query processing tool/language configuration parameters;



- d) Recovery/commit options;
- e) Consistency/locking options;
- f) Operating system and configuration parameters;
- g) Configuration parameters and options for any other software component incorporated into the pricing structure;
- h) Compiler optimization options.

**Comment:** In the event that some parameters and options are set multiple times, it must be easily discernible by an interested reader when the parameter or option was modified and what new value it received each time.

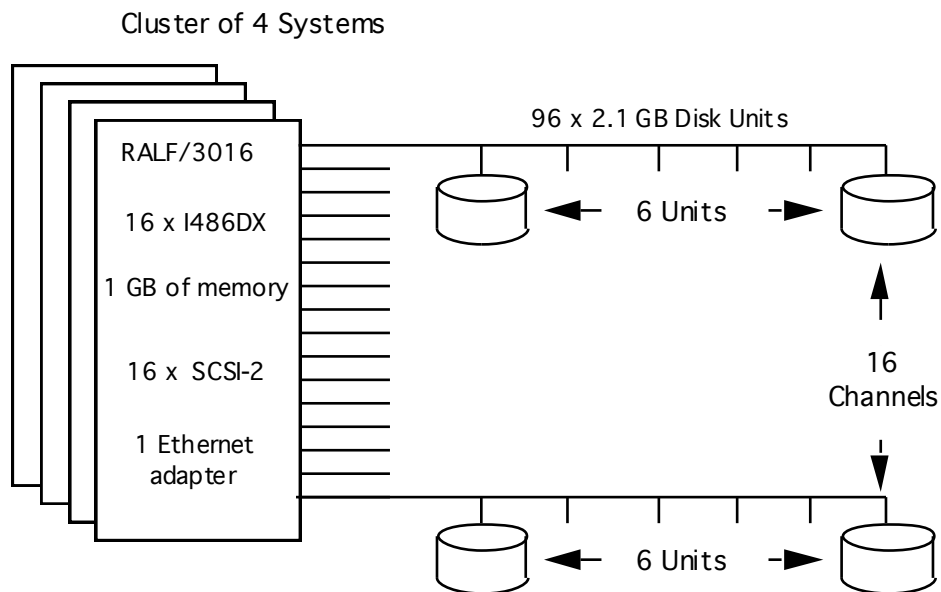
**Comment:** This requirement can be satisfied by providing a full list of all parameters and options, as long as all those that have been modified from their default values have been clearly identified and these parameters and options are only set once.

10.3.1.3 Explicit response to individual disclosure requirements specified in the body of earlier sections of this document must be provided.

10.3.1.4 Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- a) Number and type of processors (including size of L2 cache);
- b) Size of allocated memory, and any specific mapping/partitioning of memory unique to the test;
- c) Number and type of disk units (and controllers, if applicable);
- d) Number of channels or bus connections to disk units, including their protocol type;
- e) Number of LAN (e.g., Ethernet) connections, including routers, workstations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure;
- f) Type and the run-time execution location of software components (e.g., DBMS, query processing tools/languages, middleware components, software drivers, etc.).

The following sample diagram illustrates a measured benchmark configuration using Ethernet, an external driver, and four processors in the SUT. Note that this diagram does not depict or imply any optimal configuration for the TPC-DS benchmark measurement.



LAN: Ethernet using NETplus routers  
CPU: 16 x a243DX 50MHz with 256 KByte Second Level Cache  
1 gigabyte of main memory  
4 x SCSI-2 Fast Controllers  
Disk: 96 x 2.1 gigabyte SCSI-2 drives

Figure 10-1: Sample Configuration Diagram

**Comment:** Detailed diagrams for system configurations and architectures can vary widely, and it is impossible to provide exact guidelines suitable for all implementations. The intent here is to describe the system components and connections in sufficient detail to allow independent reconstruction of the measurement environment.

### 10.3.2 Clause 2- Logical Database Design Related Items

10.3.2.1 Listings must be provided for the DDL scripts and must include all table definition statements and all other statements used to set-up the test and qualification databases.

10.3.2.2 The physical organization of tables and indices within the test and qualification databases must be disclosed. If the column ordering of any table is different from that specified in Clause 2.3 or 2.4,, it must be noted.

**Comment:** The concept of physical organization includes, but is not limited to: record clustering (i.e., rows from different logical tables are co-located on the same physical data page), index clustering (i.e., rows and leaf nodes of an index to these rows are co-located on the same physical data page), and partial fill-factors (i.e., physical data pages are left partially empty even though additional rows are available to fill them).

10.3.2.3 If any directives to DDLs are used to horizontally partition tables and rows in the test and qualification databases, these directives, DDLs, and other details necessary to replicate the partitioning behavior must be disclosed.

10.3.2.4 Any replication of physical objects must be disclosed and must conform to the requirements of Clause **Error! Reference source not found..**

### 10.3.3 Clause 3 - Scaling and Database Population Related Items

10.3.3.1 The cardinality (e.g., the number of rows) of each table of the test database, as it existed at the completion of the database load (see Clause 7.1.2) must be disclosed.

10.3.3.2 The distribution of tables and logs across all media must be explicitly described using a format similar to that shown in the following example for both the tested and priced systems.

**Comment:** Detailed diagrams for layout of database tables on disks can widely vary, and it is difficult to provide exact guidelines suitable for all implementations. The intent is to provide sufficient detail to allow independent reconstruction of the test database. The figure that follows is an example of database layout descriptions and is not intended to describe any optimal layout for the TPC-DS database.

Controller	Disk Drive	Description of Content
40A	0	Operating system, root
	1	System page and swap
	2	Physical log
	3	100% of store_sales and store tables
40B	0	33% of store_sales, catalog_sales and catalog_returns tables
	1	33% of store_sales, catalog_sales and catalog_returns tables
	2	34% of store_sales, catalog_sales and catalog_returns tables
	3	100% of date_dim, time_dim and reason tables

10.3.3.3 Figure 10-2: Sample Database Layout Description

10.3.3.4

- 10.3.3.5 The mapping of database partitions/replications must be explicitly described.
- Comment:** The intent is to provide sufficient detail about partitioning and replication to allow independent reconstruction of the test database.
- 10.3.3.6 Implementations may use some form of RAID. The RAID level used must be disclosed for each device. If RAID is used in an implementation, the logical intent of its use must be disclosed. Three levels of usage are defined:
- a) Base tables only: In this case only the Base Tables (see Clause 2) are protected by any form of RAID;
  - b) Base tables and EADS: in addition to the protection of the base tables, implementations in this class must also employ RAID to protect all EADS;
  - c) Everything: implementations in this usage category must employ RAID to protect all database storage, including temporary or scratch space in addition to the base tables and EADS.
- 10.3.3.7 The version number (i.e., the major revision number, the minor revision number, and third tier number) of dsdgen must be disclosed. Any modifications to the dsdgen source code (see Appendix B:) must be disclosed. In the event that a program other than dsdgen was used to populate the database, it must be disclosed in its entirety.
- 10.3.3.8 The database load time for the test database (see Clause 7.4.3.7) must be disclosed.
- 10.3.3.9 The data storage ratio must be disclosed. It is computed by dividing the total data storage of the priced configuration (expressed in GB) by SF corresponding to the scale factor chosen for the test database as defined in Clause 3.1. The ratio must be reported to the nearest 1/100th, rounded up. For example, a system configured with 96 disks of 2.1 GB capacity for a 100GB test database has a data storage ratio of 2.02.
- Comment:** For the reporting of configured disk capacity, gigabyte (GB) is defined to be  $2^{30}$  bytes. Since disk manufacturers typically report disk size using base ten (i.e., GB =  $10^9$ ), it may be necessary to convert the advertised size from base ten to base two.
- 10.3.3.10 The details of the database load must be disclosed, including a block diagram illustrating the overall process. Disclosure of the load procedure includes all steps, scripts, input and configuration files required to completely reproduce the test and qualification databases.
- 10.3.3.11 Any differences between the configuration of the qualification database and the test database must be disclosed.
- 10.3.4 Clause 4 and 5 - Query and Data Maintenance -Related Items
- 10.3.4.1 The query language used to implement the queries must be identified (e.g., "RALF/SQL-Plus").
- 10.3.4.2 The method of verification for the random number generation must be described unless the supplied dsdgen and dsqgen were used.
- 10.3.4.3 The method used to generate values for substitution parameters must be disclosed. The version number (i.e., the major revision number, the minor revision number, and third tier number) of dsqgen must be disclosed..
- 10.3.4.4
- 10.3.4.5 The executable query text used for query validation must be disclosed along with the corresponding output data generated during the execution of the query text against the qualification database. If minor modifications have been applied to any functional query definitions or approved variants in order to obtain executable query text, these modifications must be disclosed and justified. The justification for a particular minor query modification can apply collectively to all queries for which it has been used. The output data for the power and Query Runs must be made available electronically upon request.
- Comment:** For query output of more than 10 rows, only the first 10 need to be disclosed in the FDR. The remaining rows must be made available upon request.
- 10.3.4.6 All the query substitution parameters used during the performance test must be disclosed in tabular format, along with the seeds used to generate these parameters.

- 10.3.4.7 The isolation level used to run the queries must be disclosed. If the isolation level does not map closely to one of the isolation levels defined in Clause 6.4, additional descriptive detail must be provided, along with validation tests for the alternative isolation level (see Clause 6.4.2.1)
- 10.3.4.8 All query and refresh session initialization parameters, settings and commands must be disclosed (see Clauses 7.2.2 through 7.2.7).
- 10.3.4.9 The details of how the data maintenance functions were implemented must be disclosed (including source code of any non-commercial program used).
- 10.3.4.10 Any object created in the staging area (see Clause 5.1.8 for definition and usage restrictions) used to implement the data maintenance functions must be disclosed. Also, any disk storage used for the staging area must be priced, and any mapping or virtualization of disk storage must be disclosed.
- 10.3.5 Clause 6– Data Persistence Properties Related Items
  - 10.3.5.1 The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosure of the code written to implement the ACID Transaction and Query.
- 10.3.6 Clause 7 - Performance Metrics and Execution Rules Related Items
  - 10.3.6.1 Any system activity on the SUT that takes place between the conclusion of the load test and the beginning of the performance test must be fully disclosed including listings of scripts or command logs.
  - 10.3.6.2 The details of the steps followed to implement the performance test must be disclosed.
  - 10.3.6.3 The timing intervals defined in Clause 7 must be disclosed.
  - 10.3.6.4 For each Query Run, the minimum, the 25th percentile, the median, the 75th percentile, and the maximum times for each query shall be reported.
  - 10.3.6.5 The start time and finish time for each query stream must be reported.
  - 10.3.6.6 The start time and finish time for each data maintenance function in the refresh stream must be reported for the Query Runs.
  - 10.3.6.7 The computed performance metric, related numerical quantities and the price/performance metric must be reported.
- 10.3.7 Clause 8 - SUT and Driver Implementation Related Items
  - 10.3.7.1 A detailed textual description of how the driver performs its functions, how its various components interact and any product functionalities or environmental settings on which it relies must be provided. All related source code, scripts and configuration files must be disclosed. The information provided should be sufficient for an independent reconstruction of the driver.
  - 10.3.7.2 If an implementation specific layer is used, then a detailed description of how it performs its functions, how its various components interact and any product functionalities or environmental setting on which it relies must be provided. All related source code, scripts and configuration files must be disclosed. The information provided should be sufficient for an independent reconstruction of the implementation specific layer.
  - 10.3.7.3 If profile-directed optimization as described in Clause 7.2.11 is used, such use must be disclosed. In particular, the procedure and any scripts used to perform the optimization must be disclosed.
- 10.3.8 Clause 9 - Pricing Related Items
  - 10.3.8.1 A detailed list of hardware and software used in the priced system must be reported. The rules for pricing are included in the current revision of the TPC Pricing Specification located on the TPC website (<http://www.tpc.org>).

- 10.3.8.2 The System Availability Date (see Clause 7.6.5) must be the single availability date reported on the first page of the executive summary. The full disclosure report must report Availability Dates individually for at least each of the categories for which a pricing subtotal must be. All Availability Dates required to be reported must be disclosed to a precision of 1 day, but the precise format is left to the test sponsor.

**Comment:** A test sponsor may disclose additional detail on the availability of the system's components in the Notes section of the Executive Summary and may add a footnote reference to the System Availability Date.

- 10.3.8.3 Additional Clause 7 related items may be included in the full disclosure report for each country specific priced configuration.

10.3.9 Clause 11 - Audit Related Items

- 10.3.9.1 The auditor's agency name, address, phone number, and attestation letter with a brief audit summary report indicating compliance must be included in the full disclosure report. A statement should be included specifying whom to contact in order to obtain further information regarding the audit process.

10.4 Executive Summary

The executive summary is meant to be a high level overview of a TPC-DS implementation. It should provide the salient characteristics of a benchmark execution (metrics, configuration, pricing, etc.) without the exhaustive detail found in the FDR. When the TPC-Energy optional reporting is selected by the test sponsor, the additional requirements and format of TPC-Energy related items in the executive summary are included in the TPC Energy Specification, located at [www.tpc.org](http://www.tpc.org).

The executive summary has three components:

Implementation and Cost of Ownership Overview

Pricing Spreadsheet

Numerical Quantities

10.4.1 Page Layout

Each component of the executive summary should appear on a page by itself. Each page should use a standard header and format, including

- a) 1/2 inch margins, top and bottom;
- b) 3/4 inch left margin, 1/2 inch right margin;
- c) 2 pt. frame around the body of the page. All interior lines should be 1 pt;
- d) Sponsor identification and System identification, each set apart by a 1 pt. rule, in 16-20 pt. Times Bold font.
- e) TPC-DS, TPC-Pricing, TPC-Energy (if reported), with three tier versioning (e.g., 1.2.3), and report date, separated from other header items and each other by a 1 pt. Rule, in 9-12 pt. Times font.

**Comment:** It is permissible to use or include company logos when identifying the sponsor.

**Comment:** The report date must be disclosed with a precision of 1 day. The precise format is left to the test sponsor.

**Comment:** Appendix E contains a sample executive summary. It is meant to help clarify the requirements in Clause 10.4 and is provided solely as an example.

10.4.2 Implementation Overview

## Implementation and Cost of Ownership Overview

The implementation overview page contains six sets of data, each laid out across the page as a sequence of boxes using 1 pt. rule, with a title above the required quantity. Both titles and quantities should use a 9-12 pt. Times font unless otherwise noted.

The middle portion of the page must contain two diagrams, which must be of equal size and fill out the entire space. The left diagram shows the benchmarked configuration and the right diagram shows a pie chart with the percentages of the total time and the total times for the Load Test, Throughput Run 1, and Throughput Run 2.

The next section must contain a synopsis of the SUT's major system components, including:

- Total number of nodes used/total number of processors used with their types and speeds in GHz/ total number of cores used/total number of threads used;
- Main and cache memory sizes;
- Network and I/O connectivity;
- Disk quantity and geometry.

If the implementation used a two-tier architecture, front-end and back-end systems must be detailed separately.

### 10.4.2.1 The first section contains the results that were obtained from the reported runs of the Performance test.

Title	Quantity	Precision	Units	Font
Total System Cost	3 yr. Cost of ownership (See Clause 7)	1	\$1	16-20 pt. Bold
TPC-DS Composite Query per Hour Metric	QphDS (see Clause 7.6)	0.1	QphDS@nnGB	16-20 pt. Bold
Price/Performance	\$/QphDS (see Clause 7.6.4)	1	\$/QphDS@nnGB	16-20 pt. Bold

The next section details the system configuration

Title	Quantity	Precision	Units	Font
Database Size	Raw data size of test database	1	GB	9-12 pt. Times
DBMS Manager	Brand, Software Version of DBMS used			9-12 pt. Times
Operating System	Brand, Software Version of OS used			9-12 pt. Times
Other Software	Brand, Software Version of other software components			9-12 pt. Times
System Availability Date	System Availability Date	1 day		9-12 pt. Times
Clustered Or Not	Yes/No			9-12 pt. Times

**Comment:** The Software Version must uniquely identify the orderable software product referenced in the Priced Configuration (e.g., RALF/2000 4.2.1)

### 10.4.2.2 The middle portion of the page must contain two diagrams, which must be of equal size and fill out the width of the entire space. The left diagram shows the benchmarked configuration and the right diagram shows a pie chart with the percentages of the total time and the total times for the Load Test, Throughput Run 1 and Throughput Run 2.

### 10.4.2.3 This section contains the database load and RAID information

Title	Quantity	Precision	Units	Font
Load includes backup	Yes/No	N/A	N/A	9-12 pt. Times

RAID	None / Base tables only / Explicit Auxiliary Data Structures / Everything	N/A	N/A	9-12 pt. Times
------	---	-----	-----	----------------

- 
- 10.4.2.4 The next section of the Implementation Overview shall contain a synopsis of the SUT’s major components, including:
- Node and/or processor count and speed in GHz;
  - Main and cache memory sizes;
  - Network and IO connectivity;
  - Disk quantity and geometry
  - Total mass storage in the priced system.
- If the implementation used a two-tier architecture, front-end and back-end systems should be defined separately.
- 10.4.2.5 The final section of the Implementation Overview shall contain a note stating:  
“Database Size includes only raw data (i.e., no temp, index, redundant storage space, etc.).”
- 10.4.3 Pricing Spreadsheet
- The pricing spreadsheet, required by Clause 10.4, must be reproduced in its entirety. Refer to Appendix E for a sample pricing spreadsheet.
- 10.4.4 Numerical Quantities Summary
- The Numerical Quantities Summary page contains three sets of data.
1. The first set is the number of query streams.
  2. The second set contains the Start Date, Start Time, End Date, End Time, and Elapsed Time for:
    - Database Load
    - Power Test
    - Throughput Run 1
    - Query Run1
    - Refresh sets of Throughput Run 1
    - Throughput Run 2
    - Query Run 2
    - Refresh sets of Throughput Run 2.
  3. The third set is a table which contains the information required by Clause 10.3.6.4 .
- 10.4.5 ES.xml Requirements
- The schema of the ES.xml document is defined by the XML schema document tpeds-es.xsd located on the TPC website (<http://www.tpc.org>). The ES.xml file must conform to the tpeds-es.xsd (established by XML schema validation).
- Comment:** The **Sponsor** is responsible for verifying that the ES.xml file they provide in the **Full Disclosure Report** conforms to the TPC-DS XML schema. A validation tool will be provided on the TPC web site to facilitate this verification.

Appendix G describes the structure of the XML schema, defines the individual fields, and explains how to use the schema.

## 10.5 Availability of the Full Disclosure Report

10.5.1 The full disclosure report must be readily available to the public at a reasonable charge, similar to charges for comparable documents by that test sponsor. The report must be made available when results are made public. In order to use the phrase "TPC Benchmark DS", the full disclosure report must have been submitted electronically to the TPC using the procedure described in the TPC Policies document.

10.5.2 The official full disclosure report must be available in English but may be translated to additional languages.

## 10.6 Revisions to the Full Disclosure Report

Revisions to the full disclosure documentation shall be handled as follows:

- a) Fully documented price changes can be reflected in a new published price/performance. The benchmark need not be rerun to remain compliant.
- b) Hardware or software product substitutions within the SUT, with the exception of equipment emulated as allowed under Clause 8, require the benchmark to be re-run with the new components in order to re-establish compliance. For any substitution of equipment emulated during the benchmark, a new demonstration must be provided.
- c) The revised report shall be submitted as defined in Clause 10.2.1.

**Comment:** During the normal product life cycle, problems will be uncovered that require changes, sometimes referred to as patches or updates. When the cumulative result of applied changes causes the performance metric (see Clause 7) to decrease by more than 2% from the reported value, then the test sponsor is required to re-validate the benchmark results.

- a) Fully documented price changes can be reflected in a new published price/performance.
- b) When cumulative price changes have resulted in a worsening of the reported price/performance by 2% or more the test sponsor must submit revised price/performance results to the TPC within 30 days of the effective date of the price change(s) to remain in compliance. The benchmark need not be re-run to remain in compliance.

**Comment:** The intent of this Clause is that published price/performance reflect actual current price/performance.

- a) A change in the committed availability date for the priced system can be reflected in a new published availability date.
- b) A report may be revised to add or delete Clause 9 related items for country-specific priced configurations.
- c) Full disclosure report revisions may be required for other reasons as specified in the TPC Policies and Guidelines document, and must be submitted using the mechanisms described therein.

## 10.7 Derived Results

10.7.1 TPC-DS results can be used as the basis for new TPC-DS results if and only if:

- a) The auditor ensures that the hardware and software products are the same as those used in the prior result;
- b) The auditor reviews the FDR of the new results and ensures that they match what is contained in the original sponsor's FDR;
- c) The auditor can attest to the validity of the pricing used in the new FDR.
- d) The intent of this clause is to allow a reseller of equipment from a given supplier to publish under the reseller's name a TPC-DS result already published by the supplier.



## 10.8 Supporting Files Index Table

An index for all files required by Clause 10.2.4 **Supporting Files** must be provided in the **Report**. The **Supporting Files** index is presented in a tabular format where the columns specify the following:

- The first column denotes the clause in the TPC Specification
- The second column provides a short description of the file contents
- The third column contains the path name for the file starting at the SupportingFiles directory.

If there are no **Supporting Files** provided then the description column must indicate that there is no supporting file and the path name column must be left blank.

The following table is an example of the **Supporting Files** Index Table that must be **reported** in the **Report**.

Clause	Description	Pathname
Introduction	Database Tunable Parameters	SupportingFiles/Introduction/DBtune.txt
	OS Tunable Parameters	SupportingFiles/Introduction/OSTune.txt
Clause 2	Table creation scripts	SupportingFiles/Clause2/createTables.sh
	Index creation scripts	SupportingFiles/Clause2/createIndex.sh
Clause 3	Load transaction scripts	SupportingFiles/Clause3/doLoad.sh
Clause 4		
Clause 5	Data maintenance scripts	SupportingFiles/Clause5/doRefresh.sh
Clause 6	ACID Scripts	SupportingFiles/Clause6/runACID.sh
	Output of ACID tests	SupportingFiles/Clause6/ACID.out
Clause 7		
Clause 8		
Clause 9		

## 10.9 Supporting Files

The Supporting Files contain human readable and machine executable (i.e., able to be performed by the appropriate program without modification) scripts, executables and source code that are required to recreate the benchmark Result. If there is a choice of using a GUI or a script, then the machine executable script must be provided in the Supporting Files. If no corresponding script is available for a GUI, then the Supporting Files must contain a detailed step by step description of how to manipulate the GUI.

The combination of the following rules shall allow anybody to reproduce the benchmark result.

- All software developed specifically for the benchmark must be included in the supporting files if the software was used to cover the requirements of a clause of the benchmark specification or to conduct a benchmark run with the SUT. This includes machine executable code in the form of scripts (e.g., .sql, .sh, .tcsh, .cmd, or .bat files) or source code (e.g., .cpp, .cc, .cxx, .c files). Specifically developed executables (e.g., .exe files) need to be included unless their source code has been provided in the supporting files with detailed instructions (e.g., make files) how to re-generate the executables from the source for the hardware and operating system used for the benchmark.
- References (e.g., URLs) need to be provided for all software available for general purchase or download which has NOT been developed specifically for the benchmark. The software must be available under the location provided by the references for the time the benchmark is published on the TPC website.
- All command line options used to invoke any of the above programs need to be disclosed. If a GUI is used, detailed instructions on how to navigate the GUI as used to reproduce the benchmark result need to be disclosed.

The directory structure under SupportingFiles must follow the clause numbering from the TPC-DS Standard Specification (i.e., this document). The directory name is specified by the 10.9 third level Clauses immediately preceding the fourth level **Supporting Files** reporting requirements. If there is more than one instance of one type of file, subfolders may be used for each instance. For example if multiple **Tier A** machines were used in the benchmark, there may be a folder for each **Tier A** machine.

File names should be chosen to indicate to the casual reader what is contained within the file. For example, if the requirement is to provide the scripts for all table definition statements and all other statements used to set-up the database, file names of 1, 2, 3, 4 or 5 are unacceptable. File names that include the text “tables”, “index” or “frames” should be used to convey to the reader what is being created by the script.

### 10.9.1 SupportingFiles/Introduction Directory

All scripts required to configure the hardware must be **reported** in the **Supporting Files**.

All scripts required to configure the software must be **reported** in the **Supporting Files**. This includes any **Tunable Parameters** and options which have been changed from the defaults in commercially available products, including but not limited to:

- Database tuning options.
- Recovery/commit options.
- Consistency/locking options.
- **Operating System** and application configuration parameters.
- Compilation and linkage options and run-time optimizations used to create/install applications, OS, and/or databases.
- Parameters, switches or flags that can be changed to modify the behavior of the product.

**Comment:** This requirement can be satisfied by providing a full list of all parameters and options.

### 10.9.2 SupportingFiles/Clause2 Directory

Scripts must be provided for all table definition statements and all other statements used to set-up the database. All scripts must be human readable and machine executable (i.e., able to be performed by the appropriate program without modification). All scripts are to be **reported** in the **Supporting Files**.

10.9.3 SupportingFiles/Clause3 Directory

Scripts must be provided for all **dsdgen** invocations used to populate the database with content. All scripts must be human readable and machine executable (i.e., able to be performed by the appropriate program without modification). All scripts are to be **reported** in the **Supporting Files**.

10.9.4 SupportingFiles/Clause4 Directory

The implementation of each query of the benchmark as defined per Clause 4 must be **reported** in the **Supporting Files**. This includes, but is not limited to, the code implementing the queries of this benchmark.

10.9.5 SupportingFiles/Clause5 Directory

Scripts must be provided for all steps used to maintain the database content in order to implement Clause 5. All scripts must be human readable and machine executable (i.e., able to be performed by the appropriate program without modification). All scripts are to be **reported** in the **Supporting Files**.

10.9.6 SupportingFiles/Clause6 Directory

Scripts must be provided for all steps used to validate Clause 6. All scripts must be human readable and machine executable (i.e., able to be performed by the appropriate program without modification). All scripts and the output of the scripts are to be **reported** in the **Supporting Files**.

10.9.7 SupportingFiles/Clause7 Directory

No requirements

10.9.8 SupportingFiles/Clause8 Directory

No requirements

10.9.9 SupportingFiles/Clause9 Directory

No requirements

## 11 AUDIT

This clause defines the audit requirements for TPC-DS. The auditor needs to ensure that the benchmark under audit complies with the TPC-DS specification. Rules for auditing Pricing information are included in the TPC Pricing Specification located at [www.tpc.org](http://www.tpc.org). When the TPC-Energy optional reporting is selected by the test sponsor, the rules for auditing of TPC-Energy related items are included in the TPC Energy Specification located at [www.tpc.org](http://www.tpc.org).

### 11.1 General Rules

- 11.1.1 An independent audit of the benchmark results by a TPC certified auditor is required. The term independent is defined as "the outcome of the benchmark carries no financial benefit to the auditing agency other than fees earned directly related to the audit." In addition, the auditing agency cannot have supplied any performance consulting under contract for the benchmark.

In addition, the following conditions must be met:

- a) The auditing agency cannot be financially related to the sponsor. For example, the auditing agency is financially related if it is a dependent division of the sponsor, the majority of its stock is owned by the sponsor, etc.
- b) The auditing agency cannot be financially related to any one of the suppliers of the measured/priced configuration, e.g., the DBMS supplier, the disk supplier, etc.

- 11.1.2 The auditor's attestation letter is to be made readily available to the public as part of the full disclosure report. A detailed report from the auditor is not required.

- 11.1.3 TPC-DS results can be used as the basis for new TPC-DS results if and only if:

The auditor ensures that the hardware and software products are the same as those used in the prior result;

The auditor reviews the FDR of the new results and ensures that they match what is contained in the original sponsor's FDR;

The auditor can attest to the validity of the pricing used in the new FDR.

**Comment:** The intent of this clause is to allow a reseller of equipment from a given supplier to publish under the re-seller's name a TPC-DS result already published by the supplier.

**Comment:** In the event that all conditions listed in Clause 11.1.2 are met, the auditor is not required to follow the remaining auditor's check list items from Clause 11.2.

- 11.1.4 In the event that a remote audit procedure is used in the context of a change-based audit, a remote connection to the SUT must be available for the auditor to verify selected audit items from Clause 9.2

### 11.2 Auditor's Check List

- 11.2.1 This clause defines the minimal audit checks that the auditor is required to conduct for TPC-DS. In order for the auditor to ensure that the benchmark under audit complies with the TPC-DS specification the auditor is allowed to ask for additional checks.

#### 11.2.2 Clause 2 Related Items

- 11.2.2.1 Verify that the data types used for each column are conformant. For example, verify that decimal columns can be incremented by 0.01 from -9,999,999,999.99.

- 11.2.2.2 Verify that the tables have the required list of columns.

- 11.2.2.3 Verify that the implementation rules are met by the test database.

- 11.2.2.4 Verify that the test database meets the data access transparency requirements.
- 11.2.2.5 Verify that conforming arbitrary data values can be inserted into any of the tables. Examples of verification tests include:
  - 11.2.3 Inserting a row that is a complete duplicate of an existing row except for a distinct primary key;
  - 11.2.4 Inserting a row with column values within the domain of the data type and check constraints but beyond the range of existing values.
    - 11.2.4.1 Ensure that all EADS satisfy the requirements of Clause **Error! Reference source not found.** 2.5.3
    - 11.2.4.2 Verify that the set of EADS that are present and enabled at the end of the load test are the same set that are present and enabled at the end of the performance test as required by Clause 2.5.3.6. A similar check may be performed at any point during the performance test at the discretion of the auditor. Note the method used to verify that this requirement has been met.
 

*[This auditor clause states a requirement that does not appear to be stated before (that no ADS can be created during the test). If such a requirement exists it should be stated in clause 2.]*
- 11.2.4.3 Clause 3 Related Items
  - 11.2.4.4 Verify that the qualification database is properly scaled and populated.
  - 11.2.4.5 Verify that the qualification and test databases were constructed in the same manner so that correct behavior on the qualification database is indicative of correct behavior on the test database.
  - 11.2.4.6 *Note the method used to populate the database (i.e., dsdgen or modified version of dsdgen). Note the version number (i.e., the major revision number, the minor revision number, and third tier number) of dsdgen, and the names of the dsdgen files which have been modified. Verify that the version matches the benchmark specification.*
  - 11.2.4.7 Verify that storage and processing elements that are not included in the priced configuration are physically removed or made inaccessible during the performance test using a vendor supported method.
  - 11.2.4.8 Verify that the validation data sets are proven consistent with the data loaded into the database according to clause 3.5.
  - 11.2.4.9 Verify referential integrity in the database after the initial load. Referential Integrity is a data property that can be VERIFIED BY CHECKING THAT EVERY FOREIGN KEY HAS A CORRESPONDING PRIMARY KEY.
- 11.3 Clause 4 Related Items
  - 11.3.1.1 Verify that the basis for the SQL used for each query is the functional query definition or an approved variant or meets Clause 9.2.3.2.
  - 11.3.1.2 Verify that any deviation in the SQL from either the functional query definition or an approved variant is compliant with the specified minor query modifications. Verify that minor query modifications have been applied consistently to the set of functional query definitions or approved variants used.
  - 11.3.1.3 Verify that the executable query text produces the required output when executed against the qualification database using the validation values for substitution parameters.
  - 11.3.1.4 Note the method used to generate the values for substitution parameters (i.e., dsqgen, modified version of dsqgen, other method). If dsqgen was used, note the version number (i.e., the major revision number, the minor revision number, and third tier number) of dsqgen. Verify that the version matches the benchmark specification.
  - 11.3.1.5 Verify that the generated substitution parameters are correctly generated. For each stream take 10 random queries and verify their substitution values.
  - 11.3.1.6 Verify that no aspect of the system under test, except for the database size, has changed between the demonstration of compliance against the qualification database and the execution of the reported measurements.

#### 11.4 Clause 5 Related Items

- 11.4.1.1 Verify immediately after the performance test that all EADS that were created as part of the database load are correctly maintained. This test is to be conducted with a script that performs the following two types of subtests:
  - 1. For any index measure the response time for index lookups of two keys, one that was loaded during the database load test and one that was loaded during the data maintenance run. For test Type 1 verify that both keys are in the index and verify that the query response times are not substantially different from each other, as would be the case if the index was not maintained. (e.g. having a difference of more than 50%)
  - 2. Create another instance for all non-index EADS using the same directives as used for the original EADS. Verify that the creation of the second instance does not query the original EADS. Verify that their content is logically identical.
- 11.4.1.2 Verify that the data maintenance functions are implemented according to their definition.
- 11.4.1.3 Verify that the data maintenance functions update, insert and delete the correct update data. For each data maintenance function in a random stream verify that 2 random rows have been correctly updated, inserted and deleted.
- 11.4.1.4 Verify that the transaction requirements are met by the implementation of the data maintenance functions.
- 11.4.1.5 Verify that the order of the data maintenance functions is in accordance with Clause 5.
- 11.4.1.6 Note the method used to execute database maintenance operations.
- 11.4.2 Verify that the refresh data loaded as part of each data maintenance function is in accordance with Clause 5.2.4

#### 11.5 Clause 6 Related Items

- 11.5.1.1 Verify that the required ACID properties are supported by the system under test as configured for the execution of the reported measurements.
- 11.5.1.2 If one or more of the ACID tests defined in Clause 6 were not executed, note the rationale for waiving such demonstration of support of the related ACID property.

#### 11.6 Clause 7 Related Items

- 11.6.1.1 Verify that the execution rules are followed for the Load Test, Power Test, Throughput Test 1 and Throughput Test 2.
- 11.6.1.2 Verify that the database load time is measured according to the requirements.
- 11.6.1.3 Verify that the queries are executed against the test database.
- 11.6.1.4 Verify that the query sequencing rules are followed.
- 11.6.1.5 Verify that the measurement interval for the Query Runs is measured as required.
- 11.6.1.6 Verify that the method used to measure the timing intervals is compliant.
- 11.6.1.7 Verify that the metrics are computed as required.
- 11.6.1.8 Verify that any profile-directed optimization performed by the test sponsor conforms to the requirements of Clause 7.2.
- 11.6.1.9 Verify the set of EADS that exist at the end of the load test exist and are valid and up to date at the end of the performance test by querying the meta data of the test database before the Power Test and after Throughput Test 2. If there is any doubt that the EADS are not maintained the auditor shall run additional tests.

#### 11.7 Clause 8 Related Items

- 11.7.1.1 Verify that the driver meets the requirements of Clauses 8.3.

## 11.8 Clause 9 Related Items

- 11.8.1.1 Verify that the composition of the SUT is in compliance with the Clause 9 and that its components will be commercially available products according to the current version of TPC pricing specification.
- 11.8.1.2 Note whether an implementation specific layer is used and verify its compliance with Clause 9.1.
- 11.8.1.3 Verify that all required components of the SUT are priced according to the current version of TPC pricing specification.
- 11.8.1.4 Verify that a user communication interface is included in the SUT.
- 11.8.1.5 Verify that all required maintenance is priced according to the current version of TPC pricing specification.
- 11.8.1.6 Verify that any discount used is generally available and complies according to the current version of TPC pricing specification.
- 11.8.1.7 Verify that any third-party pricing complies with the requirements of TPC pricing specification.
- 11.8.2 Verify that the pricing spreadsheet includes all hardware and software licenses, warranty coverage, and additional maintenance costs as required according to the current version of TPC pricing specification.

**Comment:** Since final pricing for new products is typically set very close to the product announcement date, the auditor is not required to verify the final pricing of the tested system.

## 11.9 Clause 10 Related Items

- 11.9.1.1 Verify that major portions of the full disclosure report are accurate and comply with the reporting requirements. This includes:
  - The executive summary;
  - The numerical quantity summary;
  - The diagrams of both measured and priced configurations;
  - The block diagram illustrating the database load process.

## Appendix A: Logical Representation of the Refresh Data Set

### A.1 Refresh Data Set DDL

The following DDL statements define a detailed structure of the flat files, generated by dsdgen, that constitute the refresh data set. The datatypes correspond to those in Clause 2.2.

Table A-1: Column definition s\_zip\_to\_gmt

Column	Datatype	NULLs	Foreign Key
zipg_zip	char(5)	N	
zipg_gmt_offset	integer	N	

Table A-2: Column definition s\_purchase\_lineitem

Column	Datatype	NULLs	Foreign Key
plin_purchase_id	identifier	N	
plin_line_number	integer	N	
plin_item_id	char(16)		i_item_id
plin_promotion_id	char(16)		p_promo_id
plin_quantity	integer		
plin_sale_price	numeric(7,2)		
plin_coupon_amt	numeric(7,2)		
plin_comment	char(100)		

Table A-3: Column definition s\_customer

Column	Datatype	NULLs	Foreign Key
cust_customer_id	identifier	N	c_customer_id
cust_salutation	char(10)		
cust_last_name	char(20)		
cust_first_name	char(20)		
cust_preferred_flag	char(1)		
cust_birth_date	char(10)		
cust_birth_country	char(20)		
cust_login_id	char(13)		
cust_email_address	char(50)		
cust_last_login_chg_date	char(10)		
cust_first_shipto_date	char(10)		
cust_first_purchase_date	char(10)		
cust_last_review_date	char(10)		
cust_primary_machine_id	char(15)		
cust_secondary_machine_id	char(15)		
cust_street_number	char(10),		
cust_suite_number	char(10)		
cust_street_name1	char(30)		
cust_street_name2	char(30)		
cust_street_type	char(15)		
cust_city	char(60)		
cust_zip	char(10)		
cust_county	char(30)		
cust_state	char(2)		
cust_country	char(20)		
cust_loc_type	char(20)		
cust_gender	char(1)		cd_gender
cust_marital_status	char(1)		cd_marital_status
cust_educ_status	char(20)		cd_education_status
cust_credit_rating	char(10)		cd_credit_rating
cust_purch_est	numeric(7,2)		cd_purchase_estimate
cust_buy_potential	char(15)		hd_buy_potential
cust_depend_cnt	integer		cd_dep_count
cust_depend_emp_cnt	integer		cd_dep_employed_count
cust_depend_college_cnt	integer		cd_dep_college_count
cust_vehicle_cnt	integer		hd_vehicle_count
cust_annual_income	numeric(9,2)		ib_lower_bound, ib_upper_bound



Table A-4: Column definition s\_purchase

Column	Datatype	NULLs	Foreign Key
purc_purchase_id	identifier	N	plin_purchase_id
purc_store_id	char(16)		s_store_id
purc_customer_id	char(16)		s_customer_id
purc_purchase_date	char(10)		d_date
purc_purchase_time	integer		t_time
purc_register_id	integer		
purc_clerk_id	integer		
purc_comment	char(100)		

Table A-5: Column definition s\_catalog\_order

Column	Datatype	NULLs	Foreign Key
cord_order_id	identifier	N	clin_order_id
cord_bill_customer_id	char(16)		c_customer_id
cord_ship_customer_id	char(16)		c_customer_id
cord_order_date	char(10)		d_date
cord_order_time	integer		t_time
cord_ship_mode_id	char(16)		sm_ship_mode_id
cord_call_center_id	char(16)		cc_call_center_id
cord_order_comments	varchar(100)		

Table A-6: Column definition s\_web\_order

Column	Datatype	NULLs	Foreign Key
word_order_id	identifier	N	wlin_order_id
word_bill_customer_id	char(16)		c_customer_id
word_ship_customer_id	char(16)		c_customer_id
word_order_date	char(10)		d_date
word_order_time	integer		t_time
word_ship_mode_id	char(16)		sm_ship_mode_id
word_web_site_id	char(16)		web_site_id
word_order_comments	char(100)		

Table A-7: Column definition s\_item

Column	Datatype	NULLs	Foreign Key
item_item_id	char(16)	N	i_item_id
item_item_description	char(200)		
item_list_price	numeric(7,2)		
item_wholesale_cost	numeric(7,2)		
item_size	char(20)		
item_formulation	char(20)		
item_color	char(20)		
item_units	char(10)		
item_container	char(10)		
item_manager_id	integer		

Table A-8: Column definition s\_catalog\_order\_lineitem

Column	Datatype	NULLs	Foreign Key
clin_order_id	identifier	N	cord_order_id
clin_line_number	integer		
clin_item_id	char(16)		i_item_id
clin_promotion_id	char(16)		p_promo_id
clin_quantity	integer		
clin_sales_price	numeric(7,2)		
clin_coupon_amt	numeric(7,2)		
clin_warehouse_id	char(16)		w_warehouse_id
clin_ship_date	char(10)		
clin_catalog_number	integer		
clin_catalog_page_number	integer		
clin_ship_cost	numeric(7,2)		

Figure A-9: Column definition s\_web\_order\_lineitem

Column	Datatype	NULLs	Foreign Key
wlin_order_id	identifier	N	word_order_id
wlin_line_number	integer	N	
wlin_item_id	char(16)		i_item_id
wlin_promotion_id	char(16)		p_promo_id
wlin_quantity	integer		
wlin_sales_price	numeric(7,2)		
wlin_coupon_amt	numeric(7,2)		
wlin_warehouse_id	char(16)		w_warehouse_id
wlin_ship_date	char(10)		d_date
wlin_ship_cost	numeric(7,2)		
wlin_web_page_id	char(16)		wp_web_page

Figure A-10: Column definition s\_store

Column	Datatype	NULLs	Foreign Key
stor_store_id	char(16)	N	s_store_id
stor_closed_date	char(10)		d_date
stor_name	char(50)		
stor_employees	integer		
stor_floor_space	integer		
stor_hours	char(20)		
stor_store_manager	char(40)		
stor_market_id	integer		
stor_geography_class	char(100)		
stor_market_manager	char(40)		
stor_tax_percentage	numeric(5,2)		

Table A-11: Column definition s\_call\_center

Column	Datatype	NULLs	Foreign Key
call_center_id	char(16)	N	cc_center_id
call_open_date	char(10)		d_date
call_closed_date	char(10)		d_date
call_center_name	char(50)		
call_center_class	char(50)		
call_center_employees	integer		
call_center_sq_ft	integer		
call_center_hours	char(20)		
call_center_manager	char(40)		
call_center_tax_percentage	numeric(7,2)		

Table A-12: Column definition s\_web\_site

Column	Datatype	NULLs	Foreign Key
wsit_web_site_id	char(16)	N	web_site_id
wsit_open_date	char(10)		d_date
wsit_closed_date	char(10)		d_date
wsit_site_name	char(50)		
wsit_site_class	char(50)		
wsit_site_manager	char(40)		
wsit_tax_percentage	decimal(5,2)		

Table A-13: Column definition s\_warehouse

Column	Datatype	NULLs	Foreign Key
wrhs_warehouse_id	char(16)	N	w_warehouse_id
wrhs_warehouse_desc	char(200)		
wrhs_warehouse_sq_ft	integer		

Table A-14: Column definition s\_web\_page

Column	Datatype	NULLs	Foreign Key
wpag_web_page_id	char(16)	N	web_page_id
wpag_create_date	char(10)		d_date

Column	Datatype	NULLs	Foreign Key
wpag_access_date	char(10)		d_date
wpag_autogen_flag	char(1)		
wpag_url	char(100)		
wpag_type	char(50)		
wpag_char_cnt	integer		
wpag_link_cnt	integer		
wpag_image_cnt	integer		
wpag_max_ad_cnt	integer		

Table A-15: Column definition s\_promotion

Column	Datatype	NULLs	Foreign Key
prom_promotion_id	char(16)	N	
prom_promotion_name	char(30)		
prom_start_date	char(10)		d_date
prom_end_date	char(10)		d_date
prom_cost	numeric(7,2)		
prom_response_target	char(1)		
prom_channel_dmail	char(1)		
prom_channel_email	char(1)		
prom_channel_catalog	char(1)		
prom_channel_tv	char(1)		
prom_channel_radio	char(1)		
prom_channel_press	char(1)		
prom_channel_event	char(1)		
prom_channel_demo	char(1)		
prom_channel_details	char(100)		
prom_purpose	char(15)		
prom_discount_active	char(1)		

Table A-16: Column definition s\_store\_returns

Column	Datatype	NULLs	Foreign Key
sret_store_id	char(16)		s_store_id
sret_purchase_id	char(16)	N	
sret_line_number	integer	N	
sret_item_id	char(16)	N	
sret_customer_id	char(16)		s_customer_id
sret_return_date	char(10)		d_date
sret_return_time	char(10)		t_time
sret_ticket_number	char(20)		
sret_return_qty	integer		
sret_return_amount	numeric(7,2)		
sret_return_tax	numeric(7,2)		
sret_return_fee	numeric(7,2)		
sret_return_ship_cost	numeric(7,2)		
sret_refunded_cash	numeric(7,2)		
sret_reversed_charge	numeric(7,2)		
sret_store_credit	numeric(7,2)		
sret_reason_id	char(16)		r_reason_id

Table A-17: Column definition s\_catalog\_returns

Column	Datatype	NULLs	Foreign Key
cret_call_center_id	char(16)		cc_call_center_id
cret_order_id	integer	N	
cret_line_number	integer	N	
cret_item_id	char(16)	N	i_item_id
cret_return_customer_id	char(16)		c_customer_id
cret_refund_customer_id	char(16)		c_customer_id
cret_return_date	char(10)		d_date
cret_return_time	char(10)		t_time
cret_return_qty	integer		
cret_return_amt	numeric(7,2)		
cret_return_tax	numeric(7,2)		

Column	Datatype	NULLs	Foreign Key
cret_return_fee	numeric(7,2)		
cret_return_ship_cost	numeric(7,2)		
cret_refunded_cash	numeric(7,2)		
cret_reversed_charge	numeric(7,2)		
cret_merchant_credit	numeric(7,2)		
cret_reason_id	char(16)		r_reason_id
cret_shipmode_id	char(16)		
cret_catalog_page_id	char(16)		
cret_warehouse_id	char(16)		

Table A-18: Column definition s\_web\_returns

Column	Datatype	NULLs	Foreign Key
wret_web_page_id	char(16)		wp_web_page_id
wret_order_id	integer	N	
wret_line_number	integer	N	
wret_item_id	char(16)	N	i_item_id
wret_return_customer_id	char(16)		c_customer_id
wret_refund_customer_id	char(16)		c_customer_id
wret_return_date	char(10)		d_date
wret_return_time	char(10)		t_time
wret_return_qty	integer		
wret_return_amt	numeric(7,2)		
wret_return_tax	numeric(7,2)		
wret_return_fee	numeric(7,2)		
wret_return_ship_cost	numeric(7,2)		
wret_refunded_cash	numeric(7,2)		
wret_reversed_charge	numeric(7,2)		
wret_account_credit	numeric(7,2)		
wret_reason_id	char(16)		r_reason_id

Table A-19: Column definition s\_inventory

Column	Datatype	NULLs	Foreign Key
invn_warehouse_id	char(16),	N	w_warehouse_id
invn_item_id	char(16),	N	i_item_id
invn_date	char(10)	N	d_date
invn_qty_on_hand	integer		

Table A-20: Column definition s\_catalog\_page

Column	Datatype	NULLs	Foreign Key
cpag_catalog_number	integer	N	
cpag_catalog_page_number	integer	N	
cpag_department	char(20)		
cpag_id	char(16)		
cpag_start_date	char(10)		d_date
cpag_end_date	char(10)		d_date
cpag_description	varchar(100)		
cpag_type	varchar(100)		

## A.2 Relationships between source schema tables

The following relationships are defined between source schema tables:

Table A-21: Column definition

Source Schema Table 1	Source Schema Table 2	Join Condition
s_purchase	s_purchase_lineitem	purc_purchase_id = plin_purchase_id
s_web_order	s_web_order_lineitem	word_order_id = wlin_order_id
s_catalog_order	s_catalog_order_lineitem	cord_order_id = clin_order_id



## Appendix B: Business Questions

**Comment:** The leading zeros in the numerical suffix used when parameters hold multiple values match the output of dsqgen. The leading zeros do not appear in the query templates.

### B.1 query1.tpl

Find customers who have returned items more than 20% more often than the average customer returns for a store in a given state for a given year.

Qualification Substitution Parameters:

- YEAR.01=2000
- STATE.01=TN
- AGGFIELD.01 = SR\_RETURN\_AMT

### B.2 query2.tpl

Report the increase of weekly web and catalog sales from one year to the next year for each week. That is, compute the increase of Monday, Tuesday, ... Sunday sales from one year to the following.

Qualification Substitution Parameters:

- YEAR.01=2001

### B.3 query3.tpl

Report the total extended sales price per item brand of a specific manufacturer for all sales in a specific month of the year.

Qualification Substitution Parameters:

- MONTH.01=11
- MANUFACT=128
- AGGC = s\_ext\_sales\_price

### B.4 query4.tpl

Find customers who spend more money via catalog than in stores. Identify preferred customers and their country of origin.

Qualification Substitution Parameters:

- YEAR.01=2001
- SELECTCONE.01=t\_s\_secyear.customer\_id,t\_s\_secyear.customer\_first\_name,t\_s\_secyear.customer\_last\_name,t\_s\_secyear.c\_preferred\_cust\_flag,t\_s\_secyear.c\_birth\_country,t\_s\_secyear.c\_login,t\_s\_secyear.c\_email\_address

### B.5 query5.tpl

Report sales, profit, return amount, and net loss in the store, catalog, and web channels for a 14-day window. Rollup results by sales channel and channel specific sales method (store for store sales, catalog page for catalog sales and web site for web sales)

Qualification Substitution Parameters:

- SALES\_DATE.01=2000-08-23
- YEAR.01=2000

B.6 query6.tpl

List all the states with at least 10 customers who during a given month bought items with the price tag at least 20% higher than the average price of items in the same category.

Qualification Substitution Parameters:

- MONTH.01=1
- YEAR.01=2001

B.7 query7.tpl

Compute the average quantity, list price, discount, and sales price for promotional items sold in stores where the promotion is not offered by mail or a special event. Restrict the results to a specific gender, marital and educational status.

Qualification Substitution Parameters:

- YEAR.01=2000
- ES.01=College
- MS.01=S
- GEN.01=M

B.8 query8.tpl

Compute the net profit of stores located in 400 Metropolitan areas with more than 10 preferred customers.

Qualification Substitution Parameters:

- |                |               |               |               |               |
|----------------|---------------|---------------|---------------|---------------|
| • ZIP.01=24128 | ZIP.81=57834  | ZIP.161=13354 | ZIP.241=15734 | ZIP.321=78668 |
| • ZIP.02=76232 | ZIP.82=62878  | ZIP.162=45375 | ZIP.242=63435 | ZIP.322=22245 |
| • ZIP.03=65084 | ZIP.83=49130  | ZIP.163=40558 | ZIP.243=25733 | ZIP.323=15798 |
| • ZIP.04=87816 | ZIP.84=81096  | ZIP.164=56458 | ZIP.244=35474 | ZIP.324=27156 |
| • ZIP.05=83926 | ZIP.85=18840  | ZIP.165=28286 | ZIP.245=24676 | ZIP.325=37930 |
| • ZIP.06=77556 | ZIP.86=27700  | ZIP.166=45266 | ZIP.246=94627 | ZIP.326=62971 |
| • ZIP.07=20548 | ZIP.87=23470  | ZIP.167=47305 | ZIP.247=53535 | ZIP.327=21337 |
| • ZIP.08=26231 | ZIP.88=50412  | ZIP.168=69399 | ZIP.248=17879 | ZIP.328=51622 |
| • ZIP.09=43848 | ZIP.89=21195  | ZIP.169=83921 | ZIP.249=15559 | ZIP.329=67853 |
| • ZIP.10=15126 | ZIP.90=16021  | ZIP.170=26233 | ZIP.250=53268 | ZIP.330=10567 |
| • ZIP.11=91137 | ZIP.91=76107  | ZIP.171=11101 | ZIP.251=59166 | ZIP.331=38415 |
| • ZIP.12=61265 | ZIP.92=71954  | ZIP.172=15371 | ZIP.252=11928 | ZIP.332=15455 |
| • ZIP.13=98294 | ZIP.93=68309  | ZIP.173=69913 | ZIP.253=59402 | ZIP.333=58263 |
| • ZIP.14=25782 | ZIP.94=18119  | ZIP.174=35942 | ZIP.254=33282 | ZIP.334=42029 |
| • ZIP.15=17920 | ZIP.95=98359  | ZIP.175=15882 | ZIP.255=45721 | ZIP.335=60279 |
| • ZIP.16=18426 | ZIP.96=64544  | ZIP.176=25631 | ZIP.256=43933 | ZIP.336=37125 |
| • ZIP.17=98235 | ZIP.97=10336  | ZIP.177=24610 | ZIP.257=68101 | ZIP.337=56240 |
| • ZIP.18=40081 | ZIP.98=86379  | ZIP.178=44165 | ZIP.258=33515 | ZIP.338=88190 |
| • ZIP.19=84093 | ZIP.99=27068  | ZIP.179=99076 | ZIP.259=36634 | ZIP.339=50308 |
| • ZIP.20=28577 | ZIP.100=39736 | ZIP.180=33786 | ZIP.260=71286 | ZIP.340=26859 |
| • ZIP.21=55565 | ZIP.101=98569 | ZIP.181=70738 | ZIP.261=19736 | ZIP.341=64457 |
| • ZIP.22=17183 | ZIP.102=28915 | ZIP.182=26653 | ZIP.262=58058 | ZIP.342=89091 |
| • ZIP.23=54601 | ZIP.103=24206 | ZIP.183=14328 | ZIP.263=55253 | ZIP.343=82136 |
| • ZIP.24=67897 | ZIP.104=56529 | ZIP.184=72305 | ZIP.264=67473 | ZIP.344=62377 |
| • ZIP.25=22752 | ZIP.105=57647 | ZIP.185=62496 | ZIP.265=41918 | ZIP.345=36233 |
| • ZIP.26=86284 | ZIP.106=54917 | ZIP.186=22152 | ZIP.266=19515 | ZIP.346=63837 |

• ZIP.27=18376	ZIP.107=42961	ZIP.187=10144	ZIP.267=36495	ZIP.347=58078
• ZIP.28=38607	ZIP.108=91110	ZIP.188=64147	ZIP.268=19430	ZIP.348=17043
• ZIP.29=45200	ZIP.109=63981	ZIP.189=48425	ZIP.269=22351	ZIP.349=30010
• ZIP.30=21756	ZIP.110=14922	ZIP.190=14663	ZIP.270=77191	ZIP.350=60099
• ZIP.31=29741	ZIP.111=36420	ZIP.191=21076	ZIP.271=91393	ZIP.351=28810
• ZIP.32=96765	ZIP.112=23006	ZIP.192=18799	ZIP.272=49156	ZIP.352=98025
• ZIP.33=23932	ZIP.113=67467	ZIP.193=30450	ZIP.273=50298	ZIP.353=29178
• ZIP.34=89360	ZIP.114=32754	ZIP.194=63089	ZIP.274=87501	ZIP.354=87343
• ZIP.35=29839	ZIP.115=30903	ZIP.195=81019	ZIP.275=18652	ZIP.355=73273
• ZIP.36=25989	ZIP.116=20260	ZIP.196=68893	ZIP.276=53179	ZIP.356=30469
• ZIP.37=28898	ZIP.117=31671	ZIP.197=24996	ZIP.277=18767	ZIP.357=64034
• ZIP.38=91068	ZIP.118=51798	ZIP.198=51200	ZIP.278=63193	ZIP.358=39516
• ZIP.39=72550	ZIP.119=72325	ZIP.199=51211	ZIP.279=23968	ZIP.359=86057
• ZIP.40=10390	ZIP.120=85816	ZIP.200=45692	ZIP.280=65164	ZIP.360=21309
• ZIP.41=18845	ZIP.121=68621	ZIP.201=92712	ZIP.281=68880	ZIP.361=90257
• ZIP.42=47770	ZIP.122=13955	ZIP.202=70466	ZIP.282=21286	ZIP.362=67875
• ZIP.43=82636	ZIP.123=36446	ZIP.203=79994	ZIP.283=72823	ZIP.363=40162
• ZIP.44=41367	ZIP.124=41766	ZIP.204=22437	ZIP.284=58470	ZIP.364=11356
• ZIP.45=76638	ZIP.125=68806	ZIP.205=25280	ZIP.285=67301	ZIP.365=73650
• ZIP.46=86198	ZIP.126=16725	ZIP.206=38935	ZIP.286=13394	ZIP.366=61810
• ZIP.47=81312	ZIP.127=15146	ZIP.207=71791	ZIP.287=31016	ZIP.367=72013
• ZIP.48=37126	ZIP.128=22744	ZIP.208=73134	ZIP.288=70372	ZIP.368=30431
• ZIP.49=39192	ZIP.129=35850	ZIP.209=56571	ZIP.289=67030	ZIP.369=22461
• ZIP.50=88424	ZIP.130=88086	ZIP.210=14060	ZIP.290=40604	ZIP.370=19512
• ZIP.51=72175	ZIP.131=51649	ZIP.211=19505	ZIP.291=24317	ZIP.371=13375
• ZIP.52=81426	ZIP.132=18270	ZIP.212=72425	ZIP.292=45748	ZIP.372=55307
• ZIP.53=53672	ZIP.133=52867	ZIP.213=56575	ZIP.293=39127	ZIP.373=30625
• ZIP.54=10445	ZIP.134=39972	ZIP.214=74351	ZIP.294=26065	ZIP.374=83849
• ZIP.55=42666	ZIP.135=96976	ZIP.215=68786	ZIP.295=77721	ZIP.375=68908
• ZIP.56=66864	ZIP.136=63792	ZIP.216=51650	ZIP.296=31029	ZIP.376=26689
• ZIP.57=66708	ZIP.137=11376	ZIP.217=20004	ZIP.297=31880	ZIP.377=96451
• ZIP.58=41248	ZIP.138=94898	ZIP.218=18383	ZIP.298=60576	ZIP.378=38193
• ZIP.59=48583	ZIP.139=13595	ZIP.219=76614	ZIP.299=24671	ZIP.379=46820
• ZIP.60=82276	ZIP.140=10516	ZIP.220=11634	ZIP.300=45549	ZIP.380=88885
• ZIP.61=18842	ZIP.141=90225	ZIP.221=18906	ZIP.301=13376	ZIP.381=84935
• ZIP.62=78890	ZIP.142=58943	ZIP.222=15765	ZIP.302=50016	ZIP.382=69035
• ZIP.63=49448	ZIP.143=39371	ZIP.223=41368	ZIP.303=33123	ZIP.383=83144
• ZIP.64=14089	ZIP.144=94945	ZIP.224=73241	ZIP.304=19769	ZIP.384=47537
• ZIP.65=38122	ZIP.145=28587	ZIP.225=76698	ZIP.305=22927	ZIP.385=56616
• ZIP.66=34425	ZIP.146=96576	ZIP.226=78567	ZIP.306=97789	ZIP.386=94983
• ZIP.67=79077	ZIP.147=57855	ZIP.227=97189	ZIP.307=46081	ZIP.387=48033
• ZIP.68=19849	ZIP.148=28488	ZIP.228=28545	ZIP.308=72151	ZIP.388=69952
• ZIP.69=43285	ZIP.149=26105	ZIP.229=76231	ZIP.309=15723	ZIP.389=25486
• ZIP.70=39861	ZIP.150=83933	ZIP.230=75691	ZIP.310=46136	ZIP.390=61547
• ZIP.71=66162	ZIP.151=25858	ZIP.231=22246	ZIP.311=51949	ZIP.391=27385
• ZIP.72=77610	ZIP.152=34322	ZIP.232=51061	ZIP.312=68100	ZIP.392=61860
• ZIP.73=13695	ZIP.153=44438	ZIP.233=90578	ZIP.313=96888	ZIP.393=58048
• ZIP.74=99543	ZIP.154=73171	ZIP.234=56691	ZIP.314=64528	ZIP.394=56910
• ZIP.75=83444	ZIP.155=30122	ZIP.235=68014	ZIP.315=14171	ZIP.395=16807
• ZIP.76=83041	ZIP.156=34102	ZIP.236=51103	ZIP.316=79777	ZIP.396=17871



- ZIP.77=12305    ZIP.157=22685    ZIP.237=94167    ZIP.317=28709    ZIP.397=35258
- ZIP.78=57665    ZIP.158=71256    ZIP.238=57047    ZIP.318=11489    ZIP.398=31387
- ZIP.79=68341    ZIP.159=78451    ZIP.239=14867    ZIP.319=25103    ZIP.399=35458
- ZIP.80=25003    ZIP.160=54364    ZIP.240=73520    ZIP.320=32213    ZIP.400=35576
- QOY.01=2
- YEAR.01=1998

#### B.9 *query9.tpl*

Categorize store sales transactions into 5 buckets according to the number of items sold. Each bucket contains the average discount amount, sales price, list price, tax, net paid, paid price including tax, or net profit..

Qualification Substitution Parameters:

- *AGGCTHEN.01* = *ss\_ext\_discount\_amt*
- *AGGCElse.01* = *ss\_net\_paid*
- *RC.01* = 74129
- *RC.02* = 122840
- *RC.03* = 56580
- *RC.04* = 10097
- *RC.05* = 165306

#### B.10 *query10.tpl*

Count the customers with the same gender, marital status, education status, purchase estimate, credit rating, dependent count, employed dependent count and college dependent count who live in certain counties and who have purchased from both stores and another sales channel during a three month time period of a given year.

Qualification Substitution Parameters:

- *YEAR.01* = 2002
- *MONTH.01* = 1
- *COUNTY.01* = Rush County
- *COUNTY.02* = Toole County
- *COUNTY.03* = Jefferson County
- *COUNTY.04* = Dona Ana County
- *COUNTY.05* = La Porte County

#### B.11 *query11.tpl*

Find customers whose increase in spending was large over the web than in stores this year compared to last year.

Qualification Substitution Parameters:

- *YEAR.01* = 2001
- *SELECTONE* = *t\_s\_secyear.customer\_preferred\_cust\_flag*

B.12 query12.tpl

Compute the revenue ratios across item classes: For each item in a list of given categories, during a 30 day time period, sold through the web channel compute the ratio of sales of that item to the sum of all of the sales in that item's class.

Qualification Substitution Parameters

- CATEGORY.01 = Sports
- CATEGORY.02 = Books
- CATEGORY.03 = Home
- SDATE.01 = 1999-02-22
- YEAR.01 = 1999

B.13 query13.tpl

Calculate the average sales quantity, average sales price, average wholesale cost, total wholesale cost for store sales of different customer types (e.g., based on marital status, education status) including their household demographics, sales price and different combinations of state and sales profit for a given year.

Qualification Substitution Parameters:

- YEAR.01 = 2001
- STATE.01 = TX
- STATE.02 = OH
- STATE.03 = TX
- STATE.04 = OR
- STATE.05 = NM
- STATE.06 = KY
- STATE.07 = VA
- STATE.08 = TX
- STATE.09 = MS
- ES.01 = Advanced Degree
- ES.02 = College
- ES.03 = 2 yr Degree
- MS.01 = M
- MS.02 = S
- MS.03 = W

B.14 query14.tpl)

This query contains multiple iterations:

**Iteration 1:** First identify items in the same brand, class and category that are sold in all three sales channels in two consecutive years. Then compute the average sales (quantity\*list price) across all sales of all three sales channels in the same three years (average sales). Finally, compute the total sales and the total number of sales rolled up for each channel, brand, class and category. Only consider sales of cross channel sales that had sales larger than the average sale.

**Iteration 2:** Based on the previous query compare December store sales.

Qualification Substitution Parameters:

- DAY.01 = 11
- YEAR.01 = 1999

B.15 query15.tpl

Report the total catalog sales for customers in selected geographical regions or who made large purchases for a given year and quarter.

Qualification Substitution Parameters:

- QOY.01 = 2
- YEAR.01 = 2001

B.16 query16.tpl

Report number of orders, total shipping costs and profits from catalog sales of particular counties and states for a given 60 day period for non-returned sales filled from an alternate warehouse.

Qualification Substitution Parameters:

- COUNTY\_E.01 = Williamson County
- COUNTY\_D.01 = Williamson County
- COUNTY\_C.01 = Williamson County
- COUNTY\_B.01 = Williamson County
- COUNTY\_A.01 = Williamson County
- STATE.01 = GA
- MONTH.01 = 2
- YEAR.01 = 2002

B.17 query17.tpl

Analyze, for each state, all items that were sold in stores in a particular quarter and returned in the next three quarters and then re-purchased by the customer through the catalog channel in the three following quarters.

Qualification Substitution Parameters:

- YEAR.01 = 2001

B.18 query18.tpl

Compute, for each county, the average quantity, list price, coupon amount, sales price, net profit, age, and number of dependents for all items purchased through catalog sales in a given year by customers who were born in a given list of six months and living in a given list of seven states and who also belong to a given gender and education demographic.

Qualification Substitution Parameters:

- MONTH.01 = 1
- MONTH.02 = 6
- MONTH.03 = 8
- MONTH.04 = 9
- MONTH.05 = 12
- MONTH.06 = 2
- STATE.01 = MS
- STATE.02 = IN
- STATE.03 = ND
- STATE.04 = OK
- STATE.05 = NM
- STATE.06 = VA

- STATE.07 = MS
- ES.01 = Unknown
- GEN.01 = F
- YEAR.01 = 1998

B.19 query19.tpl

Select the top revenue generating products bought by out of zip code customers for a given year, month and manager. Qualification Substitution Parameters

- MANAGER.01 = 8
- MONTH.01 = 11
- YEAR.01 = 1998

B.20 query20.tpl

Compute the total revenue and the ratio of total revenue to revenue by item class for specified item categories and time periods.

Qualification Substitution Parameters:

- CATEGORY.01 = Sports
- CATEGORY.02 = Books
- CATEGORY.03 = Home
- SDATE.01 = 1999-02-22
- YEAR.01 = 1999

B.21 query21.tpl

For all items whose price was changed on a given date, compute the percentage change in inventory between the 30-day period BEFORE the price change and the 30-day period AFTER the change. Group this information by warehouse.

Qualification Substitution Parameters:

- SALES\_DATE.01 = 2000-03-11
- YEAR.01 = 2000

B.22 query22.tpl

For each product name, brand, class, category, calculate the average quantity on hand. Rollup data by product name, brand, class and category.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.23 query23.tpl

This query contains multiple, related iterations:

Find frequently sold items that are items that were sold more than 4 times per day in four consecutive years. Compute the maximum store sales made by any given customer in a period of four consecutive years (same as above). Compute the best store customers as those that are in the 5<sup>th</sup> percentile of sales. Finally, compute the total sales of sales in March made by our best customers buying our most frequent items

Qualification Substitution Parameters:

- MONTH.01 = 2
- YEAR.01 = 2000
- TOPPERCENT=50

B.24 query24.tpl

This query contains multiple, related iterations:

Iteration 1: Calculate the total specified monetary value of items in a specific color for store sales transactions by customer name and store, in a specific market, from customers who currently live in their birth countries and in the neighborhood of the store, and list only those customers for whom the total specified monetary value is greater than 5% of the average value

Iteration 2: Calculate the total specified monetary value of items in a specific color and specific size for store sales transactions by customer name and store, in a specific market, from customers who currently live in their birth countries and in the neighborhood of the store, and list only those customers for whom the total specified monetary value is greater than 5% of the average value

Qualification Substitution Parameters:

- MARKET = 8
- COLOR.1 = pale
- COLOR.2 = chiffon
- AMOUNTONE = ss\_net\_paid

B.25 query25.tpl

Get all items that were

- sold in stores in a particular month and year and
- returned in the next three quarters
- re-purchased by the customer through the catalog channel in the six following months.

For these items, compute the sum of net profit of store sales, net loss of store loss and net profit of catalog . Group this information by item and store.

Qualification Substitution Parameters:

- MONTH.01 = 4
- YEAR.01 = 2001
- AGG.01 = sum

B.26 query26.tpl

Computes the average quantity, list price, discount, sales price for promotional items sold through the catalog channel where the promotion was not offered by mail or in an event for given gender, marital status and educational status.

Qualification Substitution Parameters:

- YEAR.01 = 2000
- ES.01 = College
- MS.01 = S
- GEN.01 = M

B.27 query27.tpl

For all items sold in stores located in six states during a given year, find the average quantity, average list price, average list sales price, average coupon amount for a given gender, marital status, education and customer demographic.

Qualification Substitution Parameters:

- STATE\_F.01 = TN
- STATE\_E.01 = TN
- STATE\_D.01 = TN
- STATE\_C.01 = TN
- STATE\_B.01 = TN
- STATE\_A.01 = TN
- ES.01 = College
- MS.01 = S
- GEN.01 = M
- YEAR.01 = 2002

B.28 query28.tpl

Calculate the average list price, number of non empty (null) list prices and number of distinct list prices of six different sales buckets of the store sales channel. Each bucket is defined by a range of distinct items and information about list price, coupon amount and wholesale cost.

Qualification Substitution Parameters:

- WHOLESALE COST.01=57
- WHOLESALE COST.02=31
- WHOLESALE COST.03=79
- WHOLESALE COST.04=38
- WHOLESALE COST.05=17
- WHOLESALE COST.06=7
- COUPON AMT.01=459
- COUPON AMT.02=2323
- COUPON AMT.03=12214
- COUPON AMT.04=6071
- COUPON AMT.05=836
- COUPON AMT.06=7326
- LIST PRICE.01=8
- LIST PRICE.02=90
- LIST PRICE.03=142

- LISTPRICE.04=135
- LISTPRICE.05=122
- LISTPRICE.06=154

B.29 query29.tpl

Get all items that were sold in stores in a specific month and year and which were returned in the next six months of the same year and re-purchased by the returning customer afterwards through the catalog sales channel in the following three years.

For those these items, compute the total quantity sold through the store, the quantity returned and the quantity purchased through the catalog. Group this information by item and store.

Qualification Substitution Parameters:

- MONTH.01 = 9
- YEAR.01 = 1999
- AGG.01 = 29

B.30 query30.tpl

Find customers and their detailed customer data who have returned items, which they bought on the web, for an amount that is 20% higher than the average amount a customer returns in a given state in a given time period across all items. Order the output by customer data.

Qualification Substitution Parameters:

- YEAR.01 = 2002
- STATE.01 = GA

B.31 query31.tpl

List the top five counties where the percentage growth in web sales is consistently higher compared to the percentage growth in store sales in the first three consecutive quarters for a given year.

Qualification Substitution Parameters:

- YEAR.01 = 2000
- AGG.01 = ss1.ca\_county

B.32 query32.tpl

Compute the total discounted amount for a particular manufacturer in a particular 90 day period for catalog sales whose discounts exceeded the average discount by at least 30%.

Qualification Substitution Parameters:

- CSDATE.01 = 2000-01-27
- YEAR.01 = 2000
- IMID.01 = 977

B.33 query33.tpl

What is the monthly sales figure based on extended price for a specific month in a specific year, for manufacturers in a specific category in a given time zone. Group sales by manufacturer identifier and sort output by sales amount, by channel, and give Total sales.

Qualification Substitution Parameters:

- CATEGORY.01 = Electronics
- GMT.01 = -5
- MONTH.01 = 5
- YEAR.01 = 1998

B.34 query34.tpl

Display all customers with specific buy potentials and whose dependent count to vehicle count ratio is larger than 1.2, who in three consecutive years made purchases with between 15 and 20 items in the beginning or the end of each month in stores located in 8 counties.

Qualification Substitution Parameters:

- COUNTY\_H.01 = Williamson County
- COUNTY\_G.01 = Williamson County
- COUNTY\_F.01 = Williamson County
- COUNTY\_E.01 = Williamson County
- COUNTY\_D.01 = Williamson County
- COUNTY\_C.01 = Williamson County
- COUNTY\_B.01 = Williamson County
- COUNTY\_A.01 = Williamson County
- YEAR.01 = 1999
- BPTWO.01 = unknown
- BPONE.01 = >10000



B.35 query35.tpl

For each of the customers living in the same state, having the same gender and marital status who have purchased from stores and from either the catalog or the web during a given year, display the following:

- state, gender, marital status, count of customers
- min, max, avg, count distinct of the customer's dependent count
- min, max, avg, count distinct of the customer's employed dependent count
- min, max, avg, count distinct of the customer's dependents in college count

Display / calculate the "count of customers" multiple times to emulate a potential reporting tool scenario.

Qualification Substitution Parameters:

YEAR.01 = 2002  
AGGONE = min  
AGGTWO = max  
AGGTHREE = avg

B.36 query36.tpl

Compute store sales gross profit margin ranking for items in a given year for a given list of states.\

Qualification Substitution Parameters:

- STATE\_H.01 = TN
- STATE\_G.01 = TN
- STATE\_F.01 = TN
- STATE\_E.01 = TN
- STATE\_D.01 = TN
- STATE\_C.01 = TN
- STATE\_B.01 = TN
- STATE\_A.01 = TN
- YEAR.01 = 2001

B.37 query37.tpl

List all items and current prices sold through the catalog channel from certain manufacturers in a given \$30 price range and consistently had a quantity between 100 and 500 on hand in a 60-day period.

Qualification Substitution Parameters:

- PRICE.01 = 68
- MANUFACT\_ID.01 = 677
- MANUFACT\_ID.02 = 940
- MANUFACT\_ID.03 = 694
- MANUFACT\_ID.04 = 808
- INVDATE.01 = 2000-02-01

B.38 query38.tpl

Display count of customers with purchases from all 3 channels in a given year.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.39 query39.tpl

This query contains multiple, related iterations:

Iteration 1: Calculate the coefficient of variation and mean of every item and warehouse of two consecutive months

Iteration 2: Find items that had a coefficient of variation in the first months of 1.5 or large

Qualification Substitution Parameters:

- YEAR.01 = 2001
- MONTH.01 = 1

B.40 query40.tpl

Compute the impact of an item price change on the sales by computing the total sales for items in a 30 day period before and after the price change. Group the items by location of warehouse where they were delivered from.

Qualification Substitution Parameters

- SALES\_DATE.01 = 2000-03-11
- YEAR.01 = 2000

B.41 query41.tpl

How many items do we carry with specific combinations of color, units, size and category.

Qualification Substitution Parameters

- MANUFACT.01 = 738
- SIZE.01 = medium
- SIZE.02 = extra large
- SIZE.03 = N/A
- SIZE.04 = small
- SIZE.05 = petite
- SIZE.06 = large
- UNIT.01 = Ounce
- UNIT.02 = Oz
- UNIT.03 = Bunch
- UNIT.04 = Ton
- UNIT.05 = N/A
- UNIT.06 = Dozen
- UNIT.07 = Box
- UNIT.08 = Pound
- UNIT.09 = Pallet
- UNIT.10 = Gross
- UNIT.11 = Cup
- UNIT.12 = Dram
- UNIT.13 = Each
- UNIT.14 = Tbl
- UNIT.15 = Lb
- UNIT.16 = Bundle
- COLOR.01 = powder
- COLOR.02 = khaki

- COLOR.03 = brown
- COLOR.04 = honeydew
- COLOR.05 = floral
- COLOR.06 = deep
- COLOR.07 = light
- COLOR.08 = cornflower
- COLOR.09 = midnight
- COLOR.10 = snow
- COLOR.11 = cyan
- COLOR.12 = papaya
- COLOR.13 = orange
- COLOR.14 = frosted
- COLOR.15 = forest
- COLOR.16 = ghost

B.42 query42.tpl

For each item and a specific year and month calculate the sum of the extended sales price of store transactions.

Qualification Substitution Parameters:

- MONTH.01 = 11
- YEAR.01 = 2000

B.43 query43.tpl

Report the sum of all sales from Sunday to Saturday for stores in a given data range by stores.

Qualification Substitution Parameters:

- YEAR.01 = 2000
- GMT.01 = -5

B.44 query44.tpl

List the best and worst performing products measured by net profit.

Qualification Substitution Parameters:

- NULLCOLSS.01 = ss\_addr\_sk
- STORE.01 = 4

B.45 query45.tpl

Report the total web sales for customers in specific zip codes, cities, counties or states, or specific items for a given year and quarter. .

Qualification Substitution Parameters:

- QOY.01 = 2
- YEAR.01 = 2001
- GBOBC = ca\_city

B.46 query46.tpl

Compute the per-customer coupon amount and net profit of all "out of town" customers buying from stores located in 5 cities on weekends in three consecutive years. The customers need to fit the profile of having a specific dependent count and vehicle count. For all these customers print the city they lived in at the time of purchase, the city in which the store is located, the coupon amount and net profit

Qualification Substitution Parameters:

- CITY\_E.01 = Fairview
- CITY\_D.01 = Fairview
- CITY\_C.01 = Fairview
- CITY\_B.01 = Midway
- CITY\_A.01 = Fairview
- VEHCNT.01 = 3
- YEAR.01 = 1999
- DEPCNT.01 = 4

B.47 query47.tpl

Find the item brands and categories for each store and company, the monthly sales figures for a specified year, where the monthly sales figure deviated more than 10% of the average monthly sales for the year, sorted by deviation and store. Report deviation of sales from the previous and the following monthly sales.

Qualification Substitution Parameters

- YEAR.01 = 1999
- SELECTONE = v1.i\_category, v1.i\_brand, v1.s\_store\_name, v1.s\_company\_name
- SELECTTWO = ,v1.d\_year, v1.d\_moy

B.48 query48.tpl

Calculate the total sales by different types of customers (e.g., based on marital status, education status), sales price and different combinations of state and sales profit.

Qualification Substitution Parameters:

- MS.1=M
- MS.2=D
- MS.3=S
- ES.1=4 yr Degree
- ES.2=2 yr Degree
- ES.3=College
- STATE.1=CO
- STATE.2=OH
- STATE.3=TX
- STATE.4=OR
- STATE.5=MN
- STATE.6=KY
- STATE.7=VA
- STATE.8=CA
- STATE.9=MS

B.49 Query49.tpl

Report the worst return ratios (sales to returns) of all items for each channel by quantity and currency sorted by ratio. Quantity ratio is defined as total number of sales to total number of returns. Currency ratio is defined as sum of return amount to sum of net paid.

Qualification Substitution Parameters:

- MONTH.01 = 12
- YEAR.01 = 2001

B.50 query50.tpl

For each store count the number of items in a specified month that were returned after 30, 60, 90, 120 and more than 120 days from the day of purchase.

Qualification Substitution Parameters:

- MONTH.01 = 8
- YEAR.01 = 2001

B.51 query51.tpl

Compute the count of store sales resulting from promotions, the count of all store sales and their ratio for specific categories in a particular time zone and for a given year and month.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.52 query52.tpl

Report the total of extended sales price for all items of a specific brand in a specific year and month.

Qualification Substitution Parameters

- MONTH.01=11
- YEAR.01=2000

B.53 query53.tpl

Find the ID, quarterly sales and yearly sales of those manufacturers who produce items with specific characteristics and whose average monthly sales are larger than 10% of their monthly sales.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.54 query54.tpl

Find all customers who purchased items of a given category and class on the web or through catalog in a given month and year that was followed by an in-store purchase at a store near their residence in the three consecutive months. Calculate a histogram of the revenue by these customers in \$50 segments showing the number of customers in each of these revenue generated segments.

Qualification Substitution Parameters:

- CLASS.01 = maternity
- CATEGORY.01 = Women
- MONTH.01 = 12

- YEAR.01 = 1998

B.55 query55.tpl

For a given year, month and store manager calculate the total store sales of any combination all brands.

Qualification Substitution Parameters:

- MANAGER.01 = 28
- MONTH.01 = 11
- YEAR.01 = 1999

B.56 query56.tpl

Compute the monthly sales amount for a specific month in a specific year, for items with three specific colors across all sales channels. Only consider sales of customers residing in a specific time zone. Group sales by item and sort output by sales amount.

Qualification Substitution Parameters:

- COLOR.01 = slate
- COLOR.02 = blanched
- COLOR.03 = burnished
- GMT.01 = -5
- MONTH.01 = 2
- YEAR.01 = 2001

B.57 query57.tpl

Find the item brands and categories for each call center and their monthly sales figures for a specified year, where the monthly sales figure deviated more than 10% of the average monthly sales for the year, sorted by deviation and call center. Report the sales deviation from the previous and following month.

Qualification Substitution Parameters:

- YEAR.01 = 1999
- SELECTONE = v1.i\_category, v1.i\_brand, v1.cc\_name
- SELECTTWO = ,v1.d\_year, v1.d\_moy

B.58 query58.tpl

Retrieve the items generating the highest revenue and which had a revenue that was approximately equivalent across all of store, catalog and web within the week ending a given date.

Qualification Substitution Parameters:

- SALES\_DATE.01 = 2000-01-03

B.59 query59.tpl

Report the increase of weekly store sales from one year to the next year for each store and day of the week.

Qualification Substitution Parameters:

- DMS.01 = 1212

B.60 query60.tpl

What is the monthly sales amount for a specific month in a specific year, for items in a specific category, purchased by customers residing in a specific time zone. Group sales by item and sort output by sales amount.

Qualification Substitution Parameters:

- CATEGORY.01 = Music
- GMT.01 = -5
- MONTH.01 = 9
- YEAR=1998

B.61 query61.tpl

Find the ratio of items sold with and without promotions in a given month and year. Only items in certain categories sold to customers living in a specific time zone are considered.

Qualification Substitution Parameters:

- GMT.01 = -5
- CATEGORY.01 = Jewelry
- MONTH.01 = 11
- YEAR.01 = 1998

B.62 query62.tpl

For web sales, create a report showing the counts of orders shipped within 30 days, from 31 to 60 days, from 61 to 90 days, from 91 to 120 days and over 120 days within a given year, grouped by warehouse, shipping mode and web site.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.63 query63.tpl

For a given year calculate the monthly sales of items of specific categories, classes and brands that were sold in stores and group the results by store manager. Additionally, for every month and manager print the yearly average sales of those items.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.64 query64.tpl

Find those stores that sold more cross-sales items from one year to another. Cross-sale items are items that are sold over the Internet, by catalog and in store.

Qualification Substitution Parameters:

- YEAR.01 = 1999
- PRICE.01 = 64
- COLOR.01 = purple
- COLOR.02 = burlywood
- COLOR.03 = indian
- COLOR.04 = spring
- COLOR.05 = floral

- COLOR.06 = medium

B.65 query65.tpl

In a given period, for each store, report the list of items with revenue less than 10% the average revenue for all the items in that store.

Qualification Substitution Parameters:

- DMS.01 = 1176

B.66 query66.tpl

Compute web and catalog sales and profits by warehouse. Report results by month for a given year during a given 8-hour period.

Qualification Substitution Parameters

- SALESTWO.01 = cs\_sales\_price
- SALESONE.01 = ws\_ext\_sales\_price
- NETTWO.01 = cs\_net\_paid\_inc\_tax
- NETONE.01 = ws\_net\_paid
- SMC.01 = DHL
- SMC.02 = BARIAN
- TIMEONE.01 = 30838
- YEAR.01 = 2001

B.67 query67.tpl

Find top stores for each category based on store sales in a specific year.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.68 query68.tpl

Compute the per customer extended sales price, extended list price and extended tax for "out of town" shoppers buying from stores located in two cities in the first two days of each month of three consecutive years. Only consider customers with specific dependent and vehicle counts.

Qualification Substitution Parameters:

- CITY\_B.01 = Midway
- CITY\_A.01 = Fairview
- VEH CNT.01 = 3
- YEAR.01 = 1999
- DEPCNT.01 = 4



B.69 query69.tpl

Count the customers with the same gender, marital status, education status, education status, purchase estimate and credit rating who live in certain states and who have purchased from stores but neither from the catalog nor from the web during a two month time period of a given year.

Qualification Substitution Parameters:

- STATE.01 = KY
- STATE.02 = GA
- STATE.03 = NM
- YEAR.01 = 2001
- MONTH.01 = 4

B.70 query70.tpl

Compute store sales net profit ranking by state and county for a given year and determine the five most profitable states.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.71 query71.tpl

Select the top revenue generating products, sold during breakfast or dinner time for one month managed by a given manager across all three sales channels.

Qualification Substitution Parameters:

- MANAGER.01 = 1
- MONTH.01 = 11
- YEAR.01 = 1999

B.72 query72.tpl

For each item, warehouse and week combination count the number of sales with and without promotion.

Qualification Substitution Parameters:

- BP.01 = >10000
- MS.01 = D
- YEAR.01 = 1999

B.73 query73.tpl

Count the number of customers with specific buy potentials and whose dependent count to vehicle count ratio is larger than 1 and who in three consecutive years bought in stores located in 4 counties between 1 and 5 items in one purchase. Only purchases in the first 2 days of the months are considered.

Qualification Substitution Parameters:

- COUNTY\_D.01 = Orange County
- COUNTY\_C.01 = Bronx County
- COUNTY\_B.01 = Franklin Parish
- COUNTY\_A.01 = Williamson County
- YEAR.01 = 1999
- BPTWO.01 = unknown
- BPONE.01 = >10000

B.74 query74.tpl

Display customers with both store and web sales in consecutive years for whom the increase in web sales exceeds the increase in store sales for a specified year.

Qualification Substitution Parameters:

- YEAR.01 = 2001
- AGGONE.01 = sum
- ORDERC.01 = 1 1 1

B.75 query75.tpl

For two consecutive years track the sales of items by brand, class and category.

Qualification Substitution Parameters:

- CATEGORY.01 = Books
- YEAR.02 = 2002

B.76 query76.tpl

Computes the average quantity, list price, discount, sales price for promotional items sold through the web channel where the promotion is not offered by mail or in an event for given gender, marital status and educational status.

Qualification Substitution Parameters:

- NULLCOLCS01 = cs\_ship\_addr\_sk
- NULLCOLWS.01 = ws\_ship\_customer\_sk
- NULLCOLSS.01 = ss\_store\_sk

B.77 query77.tpl

Report the total sales, returns and profit for all three sales channels for a given 30 day period. Roll up the results by channel and a unique channel location identifier.

Qualification Substitution Parameters:

- SALES\_DATE.01 = 2000-08-23

B.78 query78.tpl

Report the top customer / item combinations having the highest ratio of store channel sales to all other channel sales (minimum 2 to 1 ratio), for combinations with at least one store sale and one other channel sale. Order the output by highest ratio.

Qualification Substitution Parameters:

- YEAR.01 = 2000

B.79 query79.tpl

Compute the per customer coupon amount and net profit of Monday shoppers. Only purchases of three consecutive years made on Mondays in large stores by customers with a certain dependent count and with a large vehicle count are considered.

Qualification Substitution Parameters:

- VEH CNT.01 = 2

- YEAR.01 = 1999
- DEPCNT.01 = 6

B.80 query80.tpl

Report extended sales, extended net profit and returns in the store, catalog, and web channels for a 30 day window for items with prices larger than \$50 not promoted on television, rollup results by sales channel and channel specific sales means (store for store sales, catalog page for catalog sales and web site for web sales)

Qualification Substitution Parameters:

- SALES\_DATE.01 = 2000-08-23

B.81 query81.tpl

Find customers and their detailed customer data who have returned items bought from the catalog more than 20 percent the average customer returns for customers in a given state in a given time period. Order output by customer data.

Qualification Substitution Parameters:

- YEAR.01 = 2000
- STATE.01 = GA

B.82 query82.tpl

- Find customers who tend to spend more money (net-paid) on-line than in stores.

Qualification Substitution Parameters

- MANUFACT\_ID.01 = 129
- MANUFACT\_ID.02 = 270
- MANUFACT\_ID.03 = 821
- MANUFACT\_ID.04 = 423
- INVDATA.01 = 2000-05-25
- PRICE.01 = 62

B.83 query83.tpl

Retrieve the items with the highest number of returns where the number of returns was approximately equivalent across all store, catalog and web channels (within a tolerance of +/- 10%), within the week ending a given date.

Qualification Substitution Parameters

- RETURNED\_DATE\_THREE.01 = 2000-11-17
- RETURNED\_DATE\_TWO.01 = 2000-09-27
- RETURNED\_DATE\_ONE.01 = 2000-06-30

B.84 query84.tpl

List all customers living in a specified city, with an income between 2 values.

Qualification Substitution Parameters

- INCOME.01 = 38128
- CITY.01 = Edgewood

B.85 query85.tpl

For all web return reason calculate the average sales, average refunded cash and average return fee by different combinations of customer and sales types (e.g., based on marital status, education status, state and sales profit).

Qualification Substitution Parameters:

- YEAR.01 = 2000
- STATE.01 = IN
- STATE.02 = OH
- STATE.03 = NJ
- STATE.04 = WI
- STATE.05 = CT
- STATE.06 = KY
- STATE.07 = LA
- STATE.08 = IA
- STATE.09 = AR
- ES.01 = Advanced Degree
- ES.02 = College
- ES.03 = 2 yr Degree
- MS.01 = M
- MS.02 = S
- MS.03 = W

B.86 query86.tpl

Rollup the web sales for a given year by category and class, and rank the sales among peers within the parent, for each group compute sum of sales, location with the hierarchy and rank within the group.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.87 query87.tpl

Count how many customers have ordered on the same day items on the web and the catalog and on the same day have bought items in a store.

Qualification Substitution Parameters:

- DMS.01 = 1200

B.88 query88.tpl

How many items do we sell between pacific times of a day in certain stores to customers with one dependent count and 2 or less vehicles registered or 2 dependents with 4 or fewer vehicles registered or 3 dependents and five or less vehicles registered. In one row break the counts into sells from 8:30 to 9, 9 to 9:30, 9:30 to 10 ... 12 to 12:30

Qualification Substitution Parameters:

- STORE.01=Unknown
- HOUR.01=4
- HOUR.02=2
- HOUR.03=0

B.89 query89.tpl

Within a year list all month and combination of item categories, classes and brands that have had monthly sales larger than 0.1 percent of the total yearly sales.

Qualification Substitution Parameters:

- CLASS\_F.01 = dresses
- CAT\_F.01 = Women
- CLASS\_E.01 = birdal
- CAT\_E.01 = Jewelry
- CLASS\_D.01 = shirts
- CAT\_D.01 = Men
- CLASS\_C.01 = football
- CAT\_C.01 = Sports
- CLASS\_B.01 = stereo
- CAT\_B.01 = Electronics
- CLASS\_A.01 = computers
- CAT\_A.01 = Books
- YEAR.01 = 1999

B.90 query90.tpl

What is the ratio between the number of items sold over the internet in the morning (8 to 9am) to the number of items sold in the evening (7 to 8pm) of customers with a specified number of dependents. Consider only websites with a high amount of content.

Qualification Substitution Parameters:

- HOUR\_PM.01 = 19
- HOUR\_AM.01 = 8
- DEPCNT.01 = 6

B.91 query91.tpl

Display total returns of catalog sales by call center and manager in a particular month for male customers of unknown education or female customers with advanced degrees with a specified buy potential and from a particular time zone.

Qualification Substitution Parameters:

- YEAR.01 = 1998
- MONTH.01 = 11
- BUY\_POTENTIAL.01 = Unknown
- GMT.01 = -7

B.92 query92.tpl

Compute the total discount on web sales of items from a given manufacturer over a particular 90 day period for sales whose discount exceeded 30% over the average discount of items from that manufacturer in that period of time.

Qualification Substitution Parameters:

- IMID.01 = 350
- WSDATE.01 = 2000-01-27

B.93 query93.tpl

For a given merchandise return reason, report on customers' total cost of purchases minus the cost of returned items.

Qualification Substitution Parameters:

- Reason= reason 28

B.94 query94.tpl

Produce a count of web sales and total shipping cost and net profit in a given 60 day period to customers in a given state from a named web site for non returned orders shipped from more than one warehouse.

Qualification Substitution Parameters:

- YEAR.01 = 1999
- MONTH.01 = 2
- STATE.01 = IL

B.95 query95.tpl

Produce a count of web sales and total shipping cost and net profit in a given 60 day period to customers in a given state from a named web site for returned orders shipped from more than one warehouse.

Qualification Substitution Parameters:

- STATE.01=IL
- MONTH.01=2
- YEAR.01=1999

B.96 query96.tpl

Compute a count of sales from a named store to customers with a given number of dependents made in a specified half hour period of the day.

Qualification Substitution Parameters:

- HOUR.01 = 20
- DEPCNT.01 = 7

B.97 query97.tpl

Generate counts of promotional sales and total sales, and their ratio from the web channel for a particular item category and month to customers in a given time zone.

Qualification Substitution Parameters:

- YEAR.01 = 2000

B.98 query98.tpl

Report on items sold in a given 30 day period, belonging to the specified category.

Qualification Substitution Parameters:

- YEAR.01 = 1999
- SDATE.01 = 1999-02-22

- CATEGORY.01 = Sports
- CATEGORY.02 = Books
- CATEGORY.03 = Home

B.99 query99.tpl

For catalog sales, create a report showing the counts of orders shipped within 30 days, from 31 to 60 days, from 61 to 90 days, from 91 to 120 days and over 120 days within a given year, grouped by warehouse, call center and shipping mode.

Qualification Substitution Parameters

- DMS.01 = 1200

### Appendix C: Approved Query Variants

The following Query variants are approved. See Table 0-1 for location of Original Query Template and Approved Query Variant Templates.

Original Query Template	Approved Query Variant Template
Query18.tpl	Query18_variant1.tpl
Query27.tpl	Query27_variant1.tpl
Query36.tpl	Query36_variant1.tpl
Query51.tpl	Query51_variant1.tpl
Query70.tpl	Query70_variant1.tpl
Query77.tpl	Query77_variant1.tpl
Query80.tpl	Query80_variant1.tpl
Query86.tpl	Query86_variant1.tpl



## Appendix D: Query Ordering

The order of query templates in each query stream is determined by dsqgen. For convenience the following table displays the order of query templates for the first 21 streams. The order is the same for all scale factors.

Table 11-1 Required Query Sequences


SEQ Num	Stream Number																				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	96	83	56	89	79	73	34	70	57	15	43	95	68	23	46	51	11	86	40	90	18
2	7	32	98	5	39	66	88	53	29	60	50	31	37	81	54	38	97	61	42	47	35
3	75	30	59	52	93	4	44	92	24	62	41	17	94	99	76	74	48	49	9	25	16
4	44	92	24	62	41	17	94	99	76	74	48	49	9	25	16	27	63	8	19	58	6
5	39	66	88	53	29	60	50	31	37	81	54	38	97	61	42	47	35	67	82	55	22
6	80	84	2	7	32	98	5	39	66	88	53	29	60	50	31	37	81	54	38	97	61
7	32	98	5	39	66	88	53	29	60	50	31	37	81	54	38	97	61	42	47	35	67
8	19	58	6	80	84	2	7	32	98	5	39	66	88	53	29	60	50	31	37	81	54
9	25	16	27	63	8	19	58	6	80	84	2	7	32	98	5	39	66	88	53	29	60
10	78	77	87	72	71	65	20	64	12	1	96	83	56	89	79	73	34	70	57	15	43
11	86	40	90	18	45	3	75	30	59	52	93	4	44	92	24	62	41	17	94	99	76
12	1	96	83	56	89	79	73	34	70	57	15	43	95	68	23	46	51	11	86	40	90
13	91	13	91	13	91	13	91	13	91	13	91	13	91	13	91	13	91	13	91	13	91
14	21	36	28	69	14	21	36	28	69	14	21	36	28	69	14	21	36	28	69	14	21
15	43	95	68	23	46	51	11	86	40	90	18	45	3	75	30	59	52	93	4	44	92
16	27	63	8	19	58	6	80	84	2	7	32	98	5	39	66	88	53	29	60	50	31
17	94	99	76	74	48	49	9	25	16	27	63	8	19	58	6	80	84	2	7	32	98
18	45	3	75	30	59	52	93	4	44	92	24	62	41	17	94	99	76	74	48	49	9
19	58	6	80	84	2	7	32	98	5	39	66	88	53	29	60	50	31	37	81	54	38
20	64	12	1	96	83	56	89	79	73	34	70	57	15	43	95	68	23	46	51	11	86
21	36	28	69	14	21	36	28	69	14	21	36	28	69	14	21	36	28	69	14	21	36
22	33	85	26	10	78	77	87	72	71	65	20	64	12	1	96	83	56	89	79	73	34
23	46	51	11	86	40	90	18	45	3	75	30	59	52	93	4	44	92	24	62	41	17
24	62	41	17	94	99	76	74	48	49	9	25	16	27	63	8	19	58	6	80	84	2
25	16	27	63	8	19	58	6	80	84	2	7	32	98	5	39	66	88	53	29	60	50
26	10	78	77	87	72	71	65	20	64	12	1	96	83	56	89	79	73	34	70	57	15
27	63	8	19	58	6	80	84	2	7	32	98	5	39	66	88	53	29	60	50	31	37
28	69	14	21	36	28	69	14	21	36	28	69	14	21	36	28	69	14	21	36	28	69
29	60	50	31	37	81	54	38	97	61	42	47	35	67	82	55	22	33	85	26	10	78
30	59	52	93	4	44	92	24	62	41	17	94	99	76	74	48	49	9	25	16	27	63
31	37	81	54	38	97	61	42	47	35	67	82	55	22	33	85	26	10	78	77	87	72
32	98	5	39	66	88	53	29	60	50	31	37	81	54	38	97	61	42	47	35	67	82
33	85	26	10	78	77	87	72	71	65	20	64	12	1	96	83	56	89	79	73	34	70
34	70	57	15	43	95	68	23	46	51	11	86	40	90	18	45	3	75	30	59	52	93
35	67	82	55	22	33	85	26	10	78	77	87	72	71	65	20	64	12	1	96	83	56
36	28	69	14	21	36	28	69	14	21	36	28	69	14	21	36	28	69	14	21	36	28
37	81	54	38	97	61	42	47	35	67	82	55	22	33	85	26	10	78	77	87	72	71
38	97	61	42	47	35	67	82	55	22	33	85	26	10	78	77	87	72	71	65	20	64
39	66	88	53	29	60	50	31	37	81	54	38	97	61	42	47	35	67	82	55	22	33
40	90	18	45	3	75	30	59	52	93	4	44	92	24	62	41	17	94	99	76	74	48
41	17	94	99	76	74	48	49	9	25	16	27	63	8	19	58	6	80	84	2	7	32
42	47	35	67	82	55	22	33	85	26	10	78	77	87	72	71	65	20	64	12	1	96
43	95	68	23	46	51	11	86	40	90	18	45	3	75	30	59	52	93	4	44	92	24
44	92	24	62	41	17	94	99	76	74	48	49	9	25	16	27	63	8	19	58	6	80
45	3	75	30	59	52	93	4	44	92	24	62	41	17	94	99	76	74	48	49	9	25
46	51	11	86	40	90	18	45	3	75	30	59	52	93	4	44	92	24	62	41	17	94
47	35	67	82	55	22	33	85	26	10	78	77	87	72	71	65	20	64	12	1	96	83
48	49	9	25	16	27	63	8	19	58	6	80	84	2	7	32	98	5	39	66	88	53
49	9	25	16	27	63	8	19	58	6	80	84	2	7	32	98	5	39	66	88	53	29
50	31	37	81	54	38	97	61	42	47	35	67	82	55	22	33	85	26	10	78	77	87
51	11	86	40	90	18	45	3	75	30	59	52	93	4	44	92	24	62	41	17	94	99
52	93	4	44	92	24	62	41	17	94	99	76	74	48	49	9	25	16	27	63	8	19
53	29	60	50	31	37	81	54	38	97	61	42	47	35	67	82	55	22	33	85	26	10
54	38	97	61	42	47	35	67	82	55	22	33	85	26	10	78	77	87	72	71	65	20

SEQ Num	Stream Number																				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
55	22	33	85	26	10	78	77	87	72	71	65	20	64	12	1	96	83	56	89	79	73
56	89	79	73	34	70	57	15	43	95	68	23	46	51	11	86	40	90	18	45	3	75
57	15	43	95	68	23	46	51	11	86	40	90	18	45	3	75	30	59	52	93	4	44
58	6	80	84	2	7	32	98	5	39	66	88	53	29	60	50	31	37	81	54	38	97
59	52	93	4	44	92	24	62	41	17	94	99	76	74	48	49	9	25	16	27	63	8
60	50	31	37	81	54	38	97	61	42	47	35	67	82	55	22	33	85	26	10	78	77
61	42	47	35	67	82	55	22	33	85	26	10	78	77	87	72	71	65	20	64	12	1
62	41	17	94	99	76	74	48	49	9	25	16	27	63	8	19	58	6	80	84	2	7
63	8	19	58	6	80	84	2	7	32	98	5	39	66	88	53	29	60	50	31	37	81
64	12	1	96	83	56	89	79	73	34	70	57	15	43	95	68	23	46	51	11	86	40
65	20	64	12	1	96	83	56	89	79	73	34	70	57	15	43	95	68	23	46	51	11
66	88	53	29	60	50	31	37	81	54	38	97	61	42	47	35	67	82	55	22	33	85
67	82	55	22	33	85	26	10	78	77	87	72	71	65	20	64	12	1	96	83	56	89
68	23	46	51	11	86	40	90	18	45	3	75	30	59	52	93	4	44	92	24	62	41
69	14	21	36	28	69	14	21	36	28	69	14	21	36	28	69	14	21	36	28	69	14
70	57	15	43	95	68	23	46	51	11	86	40	90	18	45	3	75	30	59	52	93	4
71	65	20	64	12	1	96	83	56	89	79	73	34	70	57	15	43	95	68	23	46	51
72	71	65	20	64	12	1	96	83	56	89	79	73	34	70	57	15	43	95	68	23	46
73	34	70	57	15	43	95	68	23	46	51	11	86	40	90	18	45	3	75	30	59	52
74	48	49	9	25	16	27	63	8	19	58	6	80	84	2	7	32	98	5	39	66	88
75	30	59	52	93	4	44	92	24	62	41	17	94	99	76	74	48	49	9	25	16	27
76	74	48	49	9	25	16	27	63	8	19	58	6	80	84	2	7	32	98	5	39	66
77	87	72	71	65	20	64	12	1	96	83	56	89	79	73	34	70	57	15	43	95	68
78	77	87	72	71	65	20	64	12	1	96	83	56	89	79	73	34	70	57	15	43	95
79	73	34	70	57	15	43	95	68	23	46	51	11	86	40	90	18	45	3	75	30	59
80	84	2	7	32	98	5	39	66	88	53	29	60	50	31	37	81	54	38	97	61	42
81	54	38	97	61	42	47	35	67	82	55	22	33	85	26	10	78	77	87	72	71	65
82	55	22	33	85	26	10	78	77	87	72	71	65	20	64	12	1	96	83	56	89	79
83	56	89	79	73	34	70	57	15	43	95	68	23	46	51	11	86	40	90	18	45	3
84	2	7	32	98	5	39	66	88	53	29	60	50	31	37	81	54	38	97	61	42	47
85	26	10	78	77	87	72	71	65	20	64	12	1	96	83	56	89	79	73	34	70	57
86	40	90	18	45	3	75	30	59	52	93	4	44	92	24	62	41	17	94	99	76	74
87	72	71	65	20	64	12	1	96	83	56	89	79	73	34	70	57	15	43	95	68	23
88	53	29	60	50	31	37	81	54	38	97	61	42	47	35	67	82	55	22	33	85	26
89	79	73	34	70	57	15	43	95	68	23	46	51	11	86	40	90	18	45	3	75	30
90	18	45	3	75	30	59	52	93	4	44	92	24	62	41	17	94	99	76	74	48	49
91	13	91	13	91	13	91	13	91	13	91	13	91	13	91	13	91	13	91	13	91	13
92	24	62	41	17	94	99	76	74	48	49	9	25	16	27	63	8	19	58	6	80	84
93	4	44	92	24	62	41	17	94	99	76	74	48	49	9	25	16	27	63	8	19	58
94	99	76	74	48	49	9	25	16	27	63	8	19	58	6	80	84	2	7	32	98	5
95	68	23	46	51	11	86	40	90	18	45	3	75	30	59	52	93	4	44	92	24	62
96	83	56	89	79	73	34	70	57	15	43	95	68	23	46	51	11	86	40	90	18	45
97	61	42	47	35	67	82	55	22	33	85	26	10	78	77	87	72	71	65	20	64	12
98	5	39	66	88	53	29	60	50	31	37	81	54	38	97	61	42	47	35	67	82	55
99	76	74	48	49	9	25	16	27	63	8	19	58	6	80	84	2	7	32	98	5	39

## Appendix E: Sample Executive Summary

Sponsor (Optional Logo)		System Identification		TPC-DS 1.0.0 TPC-DS Pricing 1.1.0
				Report Date: November 22, 2006
Total System Cost		TPC-DS Throughput	Price / Performance	Availability Date
\$1,197,918 (USD)		3000.4 QphDRS1000GB	\$3992 \$/QphDRS1000GB	November 22, 2006
Database Size	Database Manager	Operating System	Other Software	
1000GB	Database System	OS		

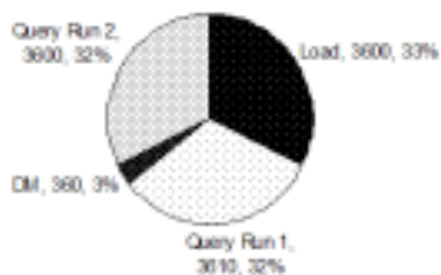


Server R (Model A) P  
30GHz CPU, 1GB (driver)

Server R (Model B) 4 x  
30GHz CPU, 10GB  
(database server)

Storage Box R (Model C)  
12x 250GB  
10K rpm SATA HDD



Query Run 2,  
3600, 32%

Load, 3600, 33%

DM, 360, 3%

Query Run 1,  
3610, 32%

Benchmarked configuration

Elapsed Time

Load Includes Backup = No	RAID = No
---------------------------	-----------

System Component	System Total	Per Node
Nodes	8	8/8
Processors/Cores/Threads: XXXX (Model A) and XXXX (Model B) Processor Y 3.4GHz, 16MB L3 Cache	8/16/16	1/2/2
Memory	64GB	8GB
OS Disk Drives: 36GB 15Krpm HDD Linn 3.20	24	3
Network Interfaces: on board 10Gb	16	2
Fibre Bus Adapters	16	2
Storage Switch	2	na
Storage Subsystem: XXX Storage Array	3	8/8
Storage Subsystem Disk Drives: 36GB 15Krpm HDD Linn 3.20	208	8/8
Total Disk Storage	7062.1 GB	8/8
* Database Subsystems only: no data (i.e., no temp, index, relational storage (log, etc.))		

Sponsor	System Identification				TPC-DS Rev. 1.0.0		
					TPC Pricing 1.1.0		
					Report Date November 22, 2006		
Description	Part Number	Price Source	Unit Price	Qty.	Extended Price	3-yr. Maint. Price	
<b>Server Hardware:</b>							
SrSr: XXX, 6u 9040 1.6GHz, 32GB Mem.	XXX-112165-100	1	\$310,900	1	\$310,900	\$34,400	
Service Processor & Server Mgmt SW	Included						
MEM: 4GB XXX Memory, 3GB DIMMs	XXX110201-ICX	1	\$3,250	8	\$8,240		
MEM: 32GB XXX Memory, 3GB DIMMs	XXX1116201-32G	1	\$11,552	6	\$68,312		
CTRL: Fibre Channel HBA, 3Gb, 2-Port PCI	XXX152321-PCX	1	\$2,135	31	\$39,186		
CTRL: Fibre Channel HBA, 4Gb, 2-Port PCI	XXX644031-PCX	1	\$1,989	3	\$5,967		
NET: 1x 36GB 10Kbps SAN (host media)	XXX112064-RDK	1	\$719	1	\$719		
DISK: 1x 36GB 10Kbps SAN (host media)	XXX4036401-32S	1	\$425	1	\$425		
FP: Monitor, 15-inch LCD, Hybrid, Msr & Cable	XXX-01P	1	\$665	1	\$665		
CBL: Ethernet, Cat5e, RMS Console, 4m	XXX600004-TBT	1	\$19	1	\$19		
CAB: 40U x 19" x 42" OpenFront Cabinet	XXX401941-F98	1	\$2,470	1	\$2,470		
CAB: 40U OpenFront Trim Kit	XXX401941-OPK	1	\$235	1	\$235		
ACC: Installation/Maintenance Materials	XXX-000-100	1	\$166	1	\$166		
ACC: XXX Call Home Service	XXX116400-CH	1	\$0	1	\$0		
Server Subtotal					\$847,620	\$34,400	
<b>Storage Hardware:</b>							
DISK: 13GB, 15Krpm, 2Gb FC *	XXX11540-CP	1	\$1,201	33	\$39,633	Spread	
CTRL: DPE, 2 RAID Card, 4GB Cache m., 6 DNs	XXX300-DPE	1	\$13,300	1	\$13,300	\$1,000	
RCK: Dual Storage Enclosure, FC 2Gb Optical I/F	XXX240-FD	1	\$5,605	1	\$5,605	\$432	
SrS MGT: Storage, Provisioning Manager	XXX300-WRK	1	\$2,135	1	\$2,135	\$1,000	
DISK: 36GB, 15Krpm, 4Gb FC *	XXX403645-4P	1	\$426	604	\$256,026	Spread	
DISK: 13GB, 15Krpm, 4Gb FC *	XXX401315-4P	1	\$396	125	\$54,500	Spread	
RCK: Storage Enclosure, FC 4Gb Optical Optical I/F	XXX4000-BAS	1	\$5,441	41	\$26,009	\$40,600	
RCK: Rack Mount Kit, Disk Enclosure	XXX4000-RMK	1	\$256	41	\$12,032		
RCK: SFP Transceiver, 4Gb LC Optical	XXX4010-1P	1	\$204	70	\$46,116	\$2,370	
CBL: FC, Optical, LC to LC Console, 8m *	XXXA2LL3M10-M	3	\$45	81	\$4,116	Spread	
PWR: Distribution Strip, 8-Socket, 200v	XXX32430-STP	1	\$404	13	\$5,252		
CAB: 40U x 19" x 42" OpenFront Cabinet	XXX401941-AUX	1	\$1,905	4	\$7,220		
CAB: 40U OpenFront Trim Kit	XXX401941-OPK	1	\$235	4	\$952		
Storage Subtotal					\$603,197	\$48,426	
<b>Server Software:</b>							
OS: XXX, 16B, 1yr Licsh.	XXX-034	1	\$35,112	1	\$35,112	\$11,434	
OS: XXX Load Sharing, 1yr.	DS5200316-TSP	1	\$3,173	2	\$6,346		
APP: XXX Database Engine	510-04170	2	\$5,315	1	\$5,315	** Inc. below	
APP: XXX Database License	359-04911	2	\$156	35	\$30,550	** Inc. below	
ACC: XXX C+	254-00170	2	\$109	1	\$109	** Inc. below	
ACC: XXX Backup	046-00556	2	\$109	1	\$109	** Inc. below	
SRVC: XXX Problem Resolution Services **		2	\$345	1		\$245	
Software Subtotal					\$98,574	\$17,669	
YAR: 10% Volume Cash Discount (Est Price when Price Source **)					(\$135,915)		
Total USD					\$1,120,625	\$77,095	
<b>Notes:</b>				<b>3-Year Cost of Ownership:</b>			
1. * 10% or more than 2 years are added in place of onsite service.				<b>Qp4 DS @ 1000GB: \$1,197,918 USD</b>			
2. HW & SW maintenance at 24 x 7 w/4 hr. max. response time for spurs.				<b>Qp4 DS @ 1000GB: 30,013.8</b>			
3. Price Source: 1 = Year Product Price, XXX Maintenance Price				<b>\$ / Qp4 DS @ 1000GB: \$3,932 USD</b>			
2 = Y/Y/Y 3 = W/W/S/Spurds							
Disclaimer: our final unit configuration is added by the Auditor, Audit & R. Inc. Ltd.							
Prices used in TPC benchmarks are not the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on exceptions about past or future purchases are not permitted. All discounts are not standard pricing policies for the listed components. For complete details, see the pricing section of the TPC benchmarks specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at <a href="mailto:pricing@tpc.org">pricing@tpc.org</a> . Thank you.							

Sponsor (Optional Logo)	System Identification	TPC-DS Rev. 1.0.0
		Report Date: November 22, 2006

Number of Query Streams: 11

Start and End Times

	Start Date	Start Time	End Date	End Time	Elapsed Time
Database Load	11/7/2008	0:00:10	11/7/2008	1:01:10	1:00:00
Power Test	11/7/2008	1:01:10	11/7/2008	1:01:10	1:00:00
Throughput Run 1	11/7/2008	2:01:10	11/7/2008	1:01:10	1:00:00
Query Run 1	11/7/2008	3:01:10	11/7/2008	1:01:10	1:00:00
Refresh Run 1	11/7/2008	4:01:10	11/7/2008	1:01:10	1:00:00
Throughput Run 2	11/7/2008	5:01:10	11/7/2008	1:01:10	1:00:00
Query Run 2	11/7/2008	6:01:10	11/7/2008	1:01:10	1:00:00
Refresh Run 2	11/7/2008	7:01:10	11/7/2008	1:01:10	1:00:00

TPC-DS Timing Intervals (in seconds)

Query	Minimum		25th Percentile		Median		75th percentile		Maximum	
	Run1	Run2	Run1	Run2	Run1	Run2	Run1	Run2	Run1	Run2
1	7.8	8.8	10.2	13.4	9.1	11.3	10.1	10.8	30.4	32.2
2	1.4	1.6	1.8	2.4	1.6	2.1	1.8	2.0	5.5	5.9
3	2.7	3.1	3.5	4.7	3.2	3.9	3.5	3.8	10.6	11.2
4	38.3	43.0	49.6	65.6	44.3	55.1	49.1	52.7	148.3	157.2
5	5.2	5.9	6.8	9.0	6.0	7.5	6.7	7.2	20.3	21.5
6	0.7	0.8	0.9	1.2	0.8	1.0	0.9	1.0	2.8	2.9
7	6.1	6.8	7.9	10.4	7.0	8.7	7.8	8.4	23.5	24.9
8	51.3	57.7	66.5	88.0	59.4	73.9	65.9	70.7	199.0	210.9
9	38.3	43.0	49.6	65.6	44.3	55.1	49.1	52.7	148.3	157.2
10	63.6	71.4	82.4	109.0	73.5	91.5	81.6	87.6	246.4	261.2
11	6.2	6.9	8.0	10.6	7.1	8.9	7.9	8.5	23.9	25.4
12	2.6	2.9	3.4	4.5	3.0	3.8	3.4	3.6	10.1	10.7
13	5.3	6.0	6.9	9.2	6.2	7.7	6.9	7.4	20.7	22.0
14	38.3	43.0	49.6	65.6	44.3	55.1	49.1	52.7	148.3	157.2
15	76.6	86.1	99.3	131.4	88.7	110.3	98.4	105.6	297.1	314.9
16	4.0	4.5	5.2	6.9	4.7	5.8	5.2	5.6	15.7	16.6
17	0.6	0.7	0.8	1.0	0.7	0.9	0.8	0.8	2.3	2.4
18	5.2	5.9	6.8	9.0	6.0	7.5	6.7	7.2	20.3	21.5
19	4.2	4.7	5.4	7.1	4.8	6.0	5.3	5.7	16.1	17.1
20	41.0	46.0	53.1	70.3	47.4	59.0	52.6	56.5	158.9	168.4
21	41.9	47.1	54.4	71.9	48.5	60.4	53.8	57.8	162.6	172.3
22	5.3	6.0	6.9	9.2	6.2	7.7	6.9	7.4	20.7	22.0
23	6.4	7.2	8.3	11.0	7.4	9.2	8.2	8.8	24.9	26.4
24	64.9	72.9	84.1	111.3	75.1	93.3	83.3	89.4	251.5	266.5
25	93.7	105.3	121.5	160.8	108.5	134.9	120.3	129.2	363.4	385.2
26	41.9	47.1	54.4	71.9	48.5	60.4	53.8	57.8	162.6	172.3
27	10.6	11.9	13.7	18.1	12.2	15.2	13.6	14.6	41.0	43.4
28	41.0	46.0	53.1	70.3	47.4	59.0	52.6	56.5	158.9	168.4
29	1.1	1.2	1.4	1.8	1.2	1.5	1.4	1.5	4.1	4.4
30	525.6	590.5	681.4	901.5	608.2	756.4	674.7	724.4	2037.4	2159.7
31	104.4	117.3	135.4	179.1	120.8	150.3	134.1	143.9	404.8	429.1
32	77.9	87.6	101.0	133.7	90.2	112.2	100.1	107.4	302.1	320.2
33	76.4	85.8	99.0	131.0	88.4	109.9	98.1	105.3	296.1	313.9
34	14.6	16.4	18.9	25.1	16.9	21.0	18.8	20.1	56.6	60.0
35	52.7	59.3	68.4	90.5	61.0	75.9	67.7	72.7	204.5	216.8

40	5.3	6.0	6.9	9.2	6.2	7.7	6.9	7.4	20.7	22.0
41	52.7	59.3	68.4	90.5	61.0	75.9	67.7	72.7	204.5	216.8
42	9.3	10.4	12.0	15.9	10.7	13.3	11.9	12.8	35.9	38.1
43	6.7	7.5	8.6	11.4	7.7	9.6	8.5	9.2	25.8	27.3
44	9.1	10.3	11.9	15.7	10.6	13.2	11.7	12.6	35.5	37.6
45	1.1	1.2	1.4	1.8	1.2	1.5	1.4	1.5	4.1	4.4
46	5.3	6.0	6.9	9.2	6.2	7.7	6.9	7.4	20.7	22.0
47	29.0	32.6	37.6	49.7	33.5	41.7	37.2	40.0	112.4	119.1
48	77.7	87.3	100.7	133.3	89.9	111.8	99.7	107.1	301.2	319.3
49	92.2	103.6	119.5	158.1	106.7	132.7	118.4	127.1	357.4	378.8
50	5.3	6.0	6.9	9.2	6.2	7.7	6.9	7.4	20.7	22.0
51	65.9	74.1	85.5	113.1	76.3	94.9	84.6	90.9	255.6	270.9
52	5.2	5.9	6.8	9.0	6.0	7.5	6.7	7.2	20.3	21.5
53	41.9	47.1	54.4	71.9	48.5	60.4	53.8	57.8	162.6	172.3
54	5.2	5.9	6.8	9.0	6.0	7.5	6.7	7.2	20.3	21.5
55	911.8	1024.4	1182.2	1564.0	1055.1	1312.2	1170.6	1256.8	3534.7	3746.8
56	5.3	6.0	6.9	9.2	6.2	7.7	6.9	7.4	20.7	22.0
57	51.7	58.1	67.0	88.6	59.8	74.4	66.3	71.2	200.3	212.4
58	51.6	57.9	66.8	88.4	59.7	74.2	66.2	71.1	199.9	211.9
59	6.4	7.2	8.3	11.0	7.4	9.2	8.2	8.8	24.9	26.4
60	5.0	5.6	6.5	8.6	5.8	7.2	6.4	6.9	19.3	20.5
61	3.8	4.3	4.9	6.5	4.4	5.5	4.9	5.2	14.7	15.6
62	52.7	59.3	68.4	90.5	61.0	75.9	67.7	72.7	204.5	216.8
63	6.4	7.2	8.3	11.0	7.4	9.2	8.2	8.8	24.9	26.4
64	53.8	60.5	69.8	92.3	62.3	77.4	69.1	74.2	208.6	221.1
65	385.4	433.0	499.7	661.1	446.0	554.6	494.8	531.2	1494.0	1583.6
66	9.1	10.3	11.9	15.7	10.6	13.2	11.7	12.6	35.5	37.6
67	105.6	118.7	136.9	181.2	122.2	152.0	135.6	145.6	409.4	434.0
68	92.4	103.8	119.8	158.5	107.0	133.0	118.7	127.4	358.3	379.8
69	91.0	102.2	118.0	156.1	105.3	131.0	116.8	125.4	352.8	373.9
70	91.1	102.4	118.1	156.3	105.4	131.1	117.0	125.6	353.2	374.4
71	5.2	5.9	6.8	9.0	6.0	7.5	6.7	7.2	20.3	21.5
72	41.9	47.1	54.4	71.9	48.5	60.4	53.8	57.8	162.6	172.3
73	5.2	5.9	6.8	9.0	6.0	7.5	6.7	7.2	20.3	21.5
74	9.1	10.3	11.9	15.7	10.6	13.2	11.7	12.6	35.5	37.6
75	5.3	6.0	6.9	9.2	6.2	7.7	6.9	7.4	20.7	22.0
76	7.8	8.8	10.2	13.4	9.1	11.3	10.1	10.8	30.4	32.2
77	51.6	57.9	66.8	88.4	59.7	74.2	66.2	71.1	199.9	211.9
78	7.8	8.8	10.2	13.4	9.1	11.3	10.1	10.8	30.4	32.2
79	5.0	5.6	6.5	8.6	5.8	7.2	6.4	6.9	19.3	20.5
80	3.8	4.3	4.9	6.5	4.4	5.5	4.9	5.2	14.7	15.6
81	5.2	5.9	6.8	9.0	6.0	7.5	6.7	7.2	20.3	21.5
82	6.4	7.2	8.3	11.0	7.4	9.2	8.2	8.8	24.9	26.4
83	53.8	60.5	69.8	92.3	62.3	77.4	69.1	74.2	208.6	221.1
84	385.4	433.0	499.7	661.1	446.0	554.6	494.8	531.2	1494.0	1583.6
85	9.1	10.3	11.9	15.7	10.6	13.2	11.7	12.6	35.5	37.6
86	105.6	118.7	136.9	181.2	122.2	152.0	135.6	145.6	409.4	434.0
87	92.4	103.8	119.8	158.5	107.0	133.0	118.7	127.4	358.3	379.8
88	42.1	47.2	54.5	72.1	48.7	60.5	54.0	58.0	163.0	172.8
89	91.1	102.4	118.1	156.3	105.4	131.1	117.0	125.6	353.2	374.4
90	5.3	6.0	6.9	9.2	6.2	7.7	6.9	7.4	20.7	22.0
91	51.7	58.1	67.0	88.6	59.8	74.4	66.3	71.2	200.3	212.4
92	5.3	6.0	6.9	9.2	6.2	7.7	6.9	7.4	20.7	22.0
93	41.0	46.0	53.1	70.3	47.4	59.0	52.6	56.5	158.9	168.4
94	4.0	4.5	5.2	6.9	4.7	5.8	5.2	5.6	15.7	16.6
95	42.2	47.4	54.7	72.3	48.8	60.7	54.1	58.1	163.5	173.3
96	41.0	46.0	53.1	70.3	47.4	59.0	52.6	56.5	158.9	168.4
97	4.2	4.7	5.4	7.1	4.8	6.0	5.3	5.7	16.1	17.1
98	552.9	621.2	716.8	948.4	639.8	795.7	709.8	762.1	2143.4	2272.0
99	76.6	86.1	99.3	131.4	88.7	110.3	98.4	105.6	297.1	314.9

## Appendix F: Tool Set Requirements

### F.1 Introduction

In addition to this document, TPC-DS relies on material that is only available electronically. While not included in the printed version of the specification, this “soft appendix” is integral to the submission of a compliant TPC-DS benchmark result.

### F.2 Availability

Need to confirm and any other legalese with TPC

The electronically available specification content may be downloaded from the TPC-DS section of the TPC web site located on the TPC website (<http://www.tpc.org>) free of charge. It is solely intended for use in conjunction with the execution of a TPC-DS benchmark. Any other use is prohibited without the express, prior written consent of the TPC.

### F.3 Compatibility

This material is maintained, versioned and revised independently of the specification itself. It is the benchmark sponsor’s responsibility to assure that any benchmark submission relies on a revision of the soft appendix that is compliant with the revision of the TPC-DS specification against which the result is being submitted.

The soft appendix includes a version number similar to that used in the specification, with a major version number, a minor version number and a third tier level, each separated by a decimal point. The major and minor revision numbers are tied to those of the TPC-DS specification with which the soft appendix is compliant. The third tier level of the soft appendix is incremented whenever the appendix itself is updated, and is independent of revision changes or updates to the specification.

A revision of the soft appendix may be used to submit a TPC-DS benchmark result provided that the major revision number of the soft appendix matches that of a specification revision that is eligible for benchmark submission;

**Comment:** The intent of this clause is to allow for the possibly lengthy tuning and preparation cycle that precedes a benchmark submission, during which a third tier revision could be released.

Benchmark sponsors are encouraged to use the most recent patch level of a given soft appendix version, as it will contain the latest clarifications and bug fixes, but any third tier level may be used to produce a compliant benchmark submission as long as the prior conditions are met.

## Appendix G: XML Schema Guide

### G.1 Overview

The schema of the ES.xml document is defined by the XML schema document tpcds-es.xsd available at <http://www.tpc.org> located on the TPC website (<http://www.tpc.org>). The ES.xml file must conform to the tpcds-es.xsd (established by XML schema validation).

### G.2 Schema Structure

An XML document conforming to the tpcds-es.xsd schema contains a single element named tpcdsResult of type RootType. The main complex types are explained in the sections below. The other types not included here can be found in tpcds-es.xsd.

### G.3 The RootType contains the following attributes:

Attribute	Type	Description
SponsorName	string	The sponsor's name.
SystemIdentification	string	The name of measured server.
SpecVersion	SpecVersionType	<i>TPC-DS SPECIFICATION VERSION NUMBER UNDER WHICH THE BENCHMARK IS PUBLISHED</i>
PricingSpecVersion	SpecVersionType	<i>TPC-PRICING SPECIFICATION VERSION NUMBER UNDER WHICH THE BENCHMARK IS PUBLISHED</i>
ReportDate	date	<i>THE DATE THAT THE RESULT IS SUBMITTED TO THE TPC.</i>
RevisionDate	date	<i>THE DATE THAT A REVISION IS SUBMITTED TO THE TPC, IF APPLICABLE; OTHERWISE OMITTED.</i>
AvailabilityDate	date	Availability Date (see TPC Pricing Specification)
TpcdsThroughput	tpcdsType	Reported Throughput in QphDS@SF (see Clause 7.6)
PricePerf	PriceType	Price/Performance Metric (\$/QphDS@SF)
Currency	CurrencyType	The currency in which the result is priced.
TotalSystemCost	PriceType	Total System Cost (see TPC Pricing Specification)
AuditorName	AuditorType	The name of the Auditor who certified the result.
Cluster	YesNoType	"Y" or "N" indicating if the result was implemented on a clustered server configuration.
AllRaidProtected	YesNoType	"Y" or "N" indicating if the result was implemented on a server with raid protection at all levels.
SchemaVersion	SchemaVersionType	The schema version, initially "1.0".

### G.4 The RootType contains the following elements:

Element	Type	Description
DBServer	DBServerType	The DBServer element contains information about the database server.
StorageSubsystem	StorageSubsystemType	The StorageSubsystem element contains information about the storage subsystem used for the benchmark run.



DatabaseLoad	DatabaseLoadRunDetailType	The DatabaseLoad element contains information about database load run performance.
PowerRun	PowerRunDetailType	The PowerRun element contains information about the Power Run performance.
QueryRun1	QueryRunDetailType	The QueryRun1 element contains information about the Query Run 1 performance.
RefreshGroup1	RefreshGroupDetailType	The RefreshRun1 element contains information about the Refresh Run 1 performance.
QueryRun2	QueryRunDetailType	The QueryRun2 element contains information about the Query Run 2 performance.
RefreshGroup2	RefreshGroupDetailType	The RefreshRun2 element contains information about the Refresh Run 2 performance.

G.5 The DBServerType contains the following attributes:

Attribute	Type	Description
DBName	string	<i>THE NAME OF THE DATABASE MANAGEMENT SYSTEM (DBMS).</i>
DBVersion	string	The version of the DBMS.
DBMiscInfo	string	<i>ANY MISCELLANEOUS INFORMATION NEEDED TO INDICATE PRECISELY WHICH VERSION OF THE DBMS WAS TESTED (E.G., "SERVICE PACK 1" OR "BUILD 1298").</i>
OSName	string	<i>THE NAME OF THE OPERATING SYSTEM (OS) ON WHICH THE DBMS WAS RUNNING.</i>
OSVersion	string	<i>THE OS VERSION.</i>
OSMiscInfo	string	<i>ANY MISCELLANEOUS INFORMATION NEEDED TO INDICATE PRECISELY WHICH VERSION OF THE OS WAS TESTED (E.G., "SERVICE PACK 1" OR "BUILD 1298").</i>
ProcessorName	string	<i>THE PROCESSOR NAME AND TYPE (E.G., "INTEL XEON 2.8 GHZ")</i>
ProcessorCount	positiveInteger	<i>THE TOTAL NUMBER OF PROCESSORS IN THE Database Server.</i>
CoreCount	positiveInteger	<i>THE TOTAL NUMBER OF CORES IN THE Database Server.</i>
ThreadCount	positiveInteger	<i>THE TOTAL NUMBER OF THREADS IN THE Database Server.</i>
Memory	decimal	<i>THE AMOUNT OF MEMORY (IN GB) CONFIGURED ON THE Database Server.</i>
InitialDBSize	positiveInteger	Initial Database Size in GB
RedundancyLevel	string	The Redundancy Level
SpindleCount	positiveInteger	Priced number of Durable Media (disks) on the Database Server.
SpindleSize	positiveInteger	Size of the priced Durable Media (disks) on the Database Server.
HostBusAdapterCount	positiveInteger	Number of host adapters in the priced system configuration

G.6 The DBServerType has the following elements:

Element	Type	Description
PerNodeHardware	PerNodeHardwareType	<i>THE HARDWARE CONFIGURATION USED FOR THE NODES. NOTE: MULTIPLE OCCURRENCES POSSIBLE.</i>

G.7 The StorageSubsystemType contains the following attributes:

Attribute	Type	Description
StorageSubsystem	string	Description or name of the storage subsystem
RaidLevel	string	RAID level used for this storage subsystem
ArrayCount	positiveInteger	Number of storage arrays used in the priced configuration
SpindleTechnology	string	Description of the durable media technology used in the priced configuration
SpindleSize	double	Size of the disk
SpindleRPM	positiveInteger	Rotations per minute of the spindles
TotalMassStorage	double	Total amount of storage provided by the storage subsystem

#### 11.9.1.2

A.1 The StorageSubsystemType contains the following elements:

Element	Type	Description
StorageArray	StorageArrayType	Description of the storage array
StorageSwitch	StorageSwitchType	Description of the storage switch

A.2 The StorageArrayType consists of the following attributes:

Attribute	Type	Description
RaidLevel	string	<i>RAID LEVEL USED SPECIFICALLY FOR THIS STORAGE ARRAY</i>
SpindleTechnology	string	<i>SPINDLE TECHNOLOGY USED SPECIFICALLY FOR THIS ARRAY</i>
SpindleCount	positiveInteger	<i>NUMBER OF SPINDLES IN THIS STORAGE ARRAY</i>
SpindleRPM	positiveInteger	<i>ROTATIONS PER MINUTE OF THE SPINDLES USED SPECIFICALLY FOR THIS STORAGE ARRAY</i>

A.3 The StorageSwitchType consists of the following attributes:

Attribute	Type	Description
StorageSwitchDescription	string	<i>DESCRIPTION OF THE STORAGE SWITCH.</i>

StorageSwitchCount	positiveInteger	<i>NUMBER OF THE STORAGE SWITCH IN THE CONFIGURATION</i>
StorageSwitchTechnology	string	<i>DESCRIPTION OF THE STORAGE SWITCH TECHNOLOGY</i>

A.4 The DatabaseLoadRunDetailType contains the following attributes:

Attribute	Type	Description
LoadTimeIncludesBackup	YesNoType	<i>'Y' OR 'N' DEPENDING ON IF THE DATABASE LOAD TIMING INCLUDES THE TIME FOR PERFORMING A BACKUP.</i>

A.5 The DatabaseLoadRunDetailType contains the following elements:

Element	Type	Description
RunTiming	RunTimingDataType	<i>THE RUN TIMING ELEMENT CONTAINS TIMING INFORMATION SUCH AS START AND END TIMES OF THE DATABASE LOAD RUN.</i>

A.6 The PowerRunDetailType contains the following elements:

Element	Type	Description
RunTiming	RunTimingDataType	<i>THE RUN TIMING ELEMENT CONTAINS TIMING INFORMATION SUCH AS START AND END TIMES OF THE POWER RUN.</i>
PowerQuery	PowerQueryDataType	<i>A QUERY ELEMENT FOR EACH QUERY IN THE BENCHMARK REPORTS ON PERFORMANCE OF THIS QUERY IN THE POWER RUN.</i>

A.7 The RefreshGroupDetailType contains the following elements:

Element	Type	Description
RunTiming	RunTimingDataType	<i>THE RUN TIMING ELEMENT CONTAINS TIMING INFORMATION SUCH AS START AND END TIMES OF A REFRESH RUN.</i>
RefreshFunction	RefreshDataType	<i>A DATA MAINTENANCE FUNCTION ELEMENT FOR EACH DATA MAINTENANCE FUNCTION PERFORMED BY EACH REFRESH RUN</i>

A.8 The QueryRunDetailType contains the following elements:

Element	Type	Description
RunTiming	RunTimingDataType	<i>THE RUN TIMING ELEMENT CONTAINS TIMING INFORMATION SUCH AS START AND END TIMES OF A QUERY RUN.</i>
Query	QueryDataType	<i>A QUERY ELEMENT FOR EACH QUERY IN THE RUN.</i>

A.9 PowerQueryDataType contains the following attributes:

Attribute	Type	Description
QueryNumber	positiveInteger	<i>QUERY NUMBER</i>
RT	RTType	<i>RUN TIME OF THE QUERY AS DEFINED BY CLAUSE 7.4.9</i>

A.10 QueryDataType contains the following attributes:

Attribute	Type	Description
QueryNumber	positiveInteger	<i>NUMBER OF THE QUERY</i>
RTMin	RTType	<i>MINIMUM RUN TIME OF THIS QUERY ACROSS ALL STREAMS IN THIS RUN</i>
RTMax	RTType	<i>MAXIMUM RUN TIME OF THIS QUERY ACROSS ALL STREAMS IN THIS RUN</i>
RTMedian	RTType	<i>MEDIAN RUN TIME OF THIS QUERY ACROSS ALL STREAMS IN THIS RUN</i>
RT25th	RTType	<i>25-PERCENTILE RUN TIME OF THIS QUERY ACROSS ALL STREAMS IN THIS RUN</i>
RT75th	RTType	<i>75-PERCENTILE RUN TIME OF THIS QUERY ACROSS ALL STREAMS IN THIS RUN</i>

A.11 RefreshDataType contains the following attributes:

Attribute	Type	Description
RefreshFunctionName	RefreshFunctionNameDataType	<i>NAME OF THE DATA MAINTENANCE FUNCTION. THE NAMES MUST MATCH ONE OF THE VALUES DEFINED IN Table 5-4.</i>
RT	RTType	<i>RUN TIME OF THIS DATA MAINTENANCE FUNCTION IN THIS RUN.</i>