TPC-DS, Taking Decision Support Benchmarking to the Next Level

Meikel Poess
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
+1-650-633-8012
Meikel.Poess@oracle.com

Bryan Smith
IBM Corporation
555 Bailey Avenue
San Jose, CA 95141
+1-408-463-2000
bfsmith@us.ibm.com

Lubor Kollar and Paul Larson
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
+1-425-882-8080
Lubork,Palarson@Microsoft.com

ABSTRACT

TPC-DS is a new decision support benchmark currently under development by the Transaction Processing Performance Council (TPC). This paper provides a brief overview of the new benchmark. The benchmark models the decision support functions of a retail product supplier, including data loading, multiple types of queries and data maintenance. The database consists of multiple snowflake schemas with shared dimension tables; data is skewed; and the query set is large. Overall, the benchmark is considerably more realistic than previous decision support benchmarks.

Categories and Subject Descriptors
H.2.4 [Database Management]: Query Processing, Relational Databases, H.2.7 [Database Management]: Database Administration, Data Warehouse and Repository, H.2.8 [Database Management]: Data Mining.

General Terms
Performance, Design, Standardization.

Keywords
Decision Support, Benchmark, Performance Evaluation, TPC, Data Warehouse

1. INTRODUCTION

The Transaction Processing Performance Council (TPC, http://www.tpc.org/), is a non-profit corporation founded in August 1988 to define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry. Full membership on the council by organization is voluntary and members are expected to participate in all aspects of the TPC's work, including development of benchmark standards and setting strategic direction. Current benchmarks cover OLTP, WEB Commerce and Decision Support (DSS) workloads. Associate memberships are also available to non-profit organizations, educational institutions, market researchers, publishers, consultants, governments, and businesses.

1.1 Previous and Existing DSS Benchmarks

In April 1995, after five years of development, the first decision support benchmark, TPC-D was approved. It differed from the existing benchmarks which focused on light and mid-weight customer oriented transactions. TPC-D represented a broad range of decision support applications that required complex, long running queries (17 queries, written in SQL-92) against large, complex data structures.

TPC-D served a very real purpose -- it improved commercial products and increased competition (69 benchmark publications from 15 different vendors). However, TPC-D had some shortcomings, the main one being that aggregate/summary structures (e.g. join indices, summary tables, materialized views, etc.) were not anticipated, and when introduced, effectively broke the benchmark. This resulted in two new benchmarks being approved, TPC-H and TPC-R in April 1999 [1,2,5].

TPC-H, an ad-hoc decision support benchmark, and TPC-R, a business reporting decision support benchmark are nearly identical to TPC-D. The difference between these two benchmarks is the prior knowledge of the workload that they assume. TPC-H represents an environment where database administrators don’t know which queries will be executed against a database system; hence, knowledge about its queries and data may not be used to optimize the DBMS system. In TPC-R, pre-knowledge of the queries is assumed and may be used for defining aggregate/summary structures [6]. This paper will focus on the new decision support benchmark TPC-DS that is currently being developed.

1.2 TPC-DS

TPC-DS, currently under development, attempts to take the best of TPC-H and TPC-R and make it more comprehensive in terms of modern day decision support. Some of the requirements for this new benchmark include the following:

- Multiple snowflake schema with shared dimensions
- Larger query set with random substitutions
- Built in data skew
- Sub-linear scaling of non-fact tables
- More realistic data maintenance
- Ad-hoc, reporting, iterative with injected wait/think time and extraction queries
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. It models the two most important components of any mature decision support system: user queries and data maintenance. The combination of queries, which convert operational facts into business intelligence, and data maintenance, which synchronizes the process of management analysis with the operational and external data sources on which it relies, constitute the core functionality of any decision support system. TPC-DS is a robust, rigorous and complete benchmark to evaluate database management systems. 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 will be explored in further detail below.

TPC-DS also highlights the system’s ability to refresh a data warehouse with new and changed data originating from the operational side of the business. Whether the business intelligence is drawn from existing operational systems or enhanced through the integration of external data sources, it must be able to respond to additions and modifications to its underlying data in a timely and cost effective manner. By focusing on the fundamental SQL-based transformations upon which all data manipulations rely, TPC-DS provides the first industry-standard evaluation of the ETL process (Extract, Transform and Load).

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

- They address complex business problems;
- They use a variety of access patterns, query phrasings, operators, and answer set constraints;
- They include a broad range of response time requirements;
- They contain both query parameters that change across query executions and scenario-based models where one query leads to the next.

The TPC-DS workload may be applied to any industry that must transform operational and external data into business intelligence. 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 models the activity of a multi-channel retailer 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.

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. The database must be able to support queries and data maintenance operations against all tables.

Some TPC benchmarks (e.g., TPC-C and TPC-W) model the operational aspect of the business environment where transactions are executed on a real time basis. Other benchmarks (i.e. TPC-H and TPC-R) address the simpler, more static model of decision support. The TPC-DS benchmark, however, models the challenges of business intelligence systems where operational data is used both to support sound business decisions in near real time and to direct long-range planning and exploration [5].

2. LOGICAL DATABASE DESIGN AND DATABASE SCALING

Most modern decision support systems (DSS) follow a star schema approach for their data model. A star schema includes a large fact table and several small dimension (lookup) tables. The fact table stores frequently added transaction data such as sales, returns and inventory changes. Each dimension table stores less frequently changed or added data supplying additional information for fact table tuples such as the customer who made a purchase. In a snowflake schema, in addition to their relation to the fact table, dimensions can have relations to other dimensions. TPC-DS’ data model consists of multiple snowflake schemas interconnected by shared dimensions [3,4].

As mentioned in the previous section, 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 imaginary retail company sells goods through the three distribution channels, store, catalog and Internet (web). An inventory fact table is shared between the catalog and the Web channel. Figure 1 demonstrates the relationships of the seven fact tables (please note that the dimensions are omitted).

![Figure 1: Logical Database Design (Fact Tables only)](image)

Each distribution channel consists of a pair of fact tables, modeling sales and return transactions. The number of dimensions emerging from each distribution channel (pair of sales and returns) is different. Catalog has 11 dimensions, Web Sales has 11 dimensions and Store has 8 dimensions. However, most dimensions are shared among distribution channels limiting the total number of dimensions to 15. In addition to the tables for the three distribution channels, there exists an inventory fact table. Inventory is related only to the web and catalog distribution channels – stores track their own inventory.

TPC-DS is moving to a snowflake schema for a number of reasons. It reflects the most challenging database environment that database customers are using today for their DSS. As a common extension to the star schema it separates static data in the outlying dimension tables from the more dynamic data in the inner dimension tables and the fact tables. A snowflake schema implies a unique stress on the database optimizer to create the most efficient execution plan while simultaneously providing for efficient joins through surrogate keys and easy query development.

The preliminary scaling rules for the benchmark are modeled after the rules used in TPC-H and TPC-R. The benchmark defines a set of discrete scaling points (scale factors [GB]), based on the approximate size of the raw data produced by the TPC-DS data generator. These scale factors are currently 1, 10, 100, 1000, 10000, 100000. The TPC provides tools to generate the initial load and refresh data set (see Data Maintenance Process).

For each scale factor the majority (about 99%) of all data is situated in fact tables. The three main fact tables (Store Sales, Web Returns, and Catalog Returns)
Catalog Sales and Web Sales) scale linearly with the scale factor (see Figure 2). The return table for stores constitutes 1 percent of its sales table while catalog and web returns constitute 5 percent of their sales tables. The Inventory Fact Table scales according to the number of items and warehouses.

### Fact Table Scaling

<table>
<thead>
<tr>
<th>Scale Factor</th>
<th># Rows in Fact Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00E+05</td>
<td>Store Sales</td>
</tr>
<tr>
<td>1.00E+06</td>
<td>Catalog Sales</td>
</tr>
<tr>
<td>1.00E+07</td>
<td>Web Sales</td>
</tr>
<tr>
<td>1.00E+08</td>
<td>Inventory</td>
</tr>
</tbody>
</table>

Figure 2: Fact Table Scaling

TPC-DS supports static and non-static dimensions. Non-static Dimension tables scale sub-linearly such as the store and item dimension. The date dimension is an example of a static dimension (see Figure 3).

### Dimension Table Scaling

<table>
<thead>
<tr>
<th>Scale Factor</th>
<th># Rows in Dimension Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stores</td>
</tr>
<tr>
<td>10</td>
<td>Date</td>
</tr>
<tr>
<td>100</td>
<td>Item</td>
</tr>
</tbody>
</table>

Figure 3: Dimension Table Scaling

One of the deficiencies of the previous TPC decision support benchmarks was the uniform distribution of data in the schema. TPC-DS defines data distributions that are both more realistic and provide additional challenges for DBMS optimizers while still preserving the ability for multiple query sets to be comparable. For instance, stores sales only occur between 9:00am and 9:00pm, and in that period, there are two peaks -- one during the noon hour, and another in the evening at 7:00pm. Additionally, store sales slowly rise from October to a peak in December of each year. For web and catalog subject areas, the distribution of sales during a day does not have any time bounds, but sales fluctuate by time of day.

3. EXECUTION MODEL

Execution of the benchmark consists of four steps. The first step is a Load Test followed by a first Query Run, a Data Maintenance Run, and it concludes with a second Query Run. The sequence of queries to be executed in the first and second query runs is identical. Repeated execution of the same queries during the second query run demonstrates the effect of the Data Maintenance step on the overall system performance.

3.1 Load Test

The Load Test demonstrates the bulk load performance of the system. The data is presented to the system in the form of flat files that resemble the star schema of the data warehouse (see section on logical database design and database scaling). Therefore no data transformations other than basic character to number and date transformations are required. The Load Test begins when the first character is read from any of the flat files. It includes all activities required to bring the system to the state that immediately precedes the beginning of the first query run such as create the tables, load data, create auxiliary structures, define and validate constraints, gather statistics, configure the system as it will be during the Query and Data Maintenance Runs, and ensure that the test database meets the ACID requirements including synchronizing loaded data on RAID devices and taking of a backup of the database, when necessary.

3.2 Query Run

The Query Run measures the system’s ability to use all of its resources to satisfy concurrent users in the shortest possible time. It executes a set of scenarios. Each scenario consists of one or several queries with possible think times. In the case of a reporting, ad-hoc DSS or data extraction query, a scenario is usually a single query, while an iterative OLAP scenario often comprises several queries. In the case of user interaction, the sequence of queries represents a user issuing several related queries, usually the subsequent query adding, removing or modifying predicates from the previous one. Think times can be included between queries in one scenario. It is intended to model the time that human operators require to analyze a query result before the next query is submitted.

Initial portion of the Query Run sequence of scenarios:

\[
[q_1],[q_2],[q_3,t_3,q_4,t_4,q_5],[q_6],[q_7],[q_8],[q_9],[q_{10},q_{11},q_{12}],[q_{13}],[q_{14},t_{14},q_{15}],[q_{16}],[q_{17}] \ldots
\]

Input sequence of scenarios prepared for execution

\[
[q_1] \quad [q_2] \quad [q_3,t_{13},q_{14},t_{14},q_{15},q_{16},q_{17}] \quad [q_6] \quad [q_7] \quad [q_8]
\]

Completed Scenario

Currently executed scenarios

Figure 4: Query Run with 6 execution streams

Scenarios are arranged in a single input sequence and they are submitted to several execution streams. Each execution stream represents one active session on the system. The next scenario is submitted for execution to the session that has just
finished processing a scenario. All sessions process their scenarios concurrently until the input sequence is exhausted and all queries return all results. Once the Query Run starts execution, the session may be idle only if there are no more scenarios on the input sequence. A session is not considered idle if it "executes" benchmark prescribed wait time between two queries.

TPC-DS defines a minimal number of execution streams for each possible benchmark scale. Figure 4 shows a possible state of a Query Run with 6 execution streams. Individual queries are $q_1,q_2,...$, think times $t_1,t_2,...$, and the scenarios are in square brackets.

The time of a Query Run is measured from the time the first scenario is submitted for execution to the time when all scenarios complete execution. Execution of a scenario is complete when all result sets are presented to the session executing the scenario. The primary performance metric of the benchmark is $Qph-DS@SF$, the effective query throughput of the benchmarked configuration, defined as:

$$QphDS@SF = \frac{N}{T_{query} + T_{update}}$$

Where $N$ is the query set size for SF, $T_{query}$ is the total elapsed time to complete query run 1 and query run 2 and $T_{update}$ is the total elapsed time to complete the refresh run.

### 3.3 Data Maintenance

The Data Maintenance Run can start any time after the first Query Run and must end before the start of the second Query Run. Its elapsed time is measured from the moment the refresh data set is accessed and is complete when all required updates on the database tables are complete. The end time of the refresh is identical with the start time of the second query run.

### 4. QUERY MODEL

A sophisticated decision support system must provide for a diverse user population simultaneously supporting many different user classes. For example, one user class may submit OLAP queries. This user class would likely have a relatively high user count with minimal think time and a tight response time constraint; there is likely to be a high correlation between one query and the next with a preference for answer sets of 100 result rows or fewer. Another user class may tend toward large answer sets, a small user population, no linkage between queries, and complex query syntax and long think times – something more akin to a knowledge worker performing data mining.

The TPC-DS benchmark models several classes of user queries that represent different kinds of database activities of a DSS system. The classes are reporting, ad-hoc DSS, interactive OLAP and data extraction queries. Queries are written in SQL 99 with OLAP amendment. TPC-DS defines the following distribution of the four query classes.

<table>
<thead>
<tr>
<th>Query Class</th>
<th>Percent of Templates</th>
<th>Number of Templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reporting</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Ad-Hoc DSS</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>Iterative OLAP</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>Data Extraction</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>

The large query set needed for TPC-DS (about 500 queries) is generated from query templates by a query generator. The query generator provides for scalar, operator and predicate substitution. Scalar substitution is defined to be the replacement of a substitution tag with one of the prescribed substitution values. The values can be either a constant string or a numeric value.

Operator substitution is a substitution in which the tag is replaced by one of a set of equivalent operators. During predicate substitution, additional where-clause predicates are added to the query text, possibly duplicating an existing predicate. Predicate introduction can also be used to introduce tautologies into a query. A special template language has been defined to direct the query generator in the different substitutions. Due to space constraints we are not able to describe the template language in this paper.

#### 4.1 Reporting Query Class

Similar to TPC-H and TPC-R, TPC-DS benchmark has very complex reporting queries. This query class captures the "reporting" nature of a DSS system in that it includes queries that are executed periodically to answer business questions. Queries in this class tend to be largely static, meaning that variations from one use of a query template to the next are likely to be limited to replacement of scalar values. The limited variation of these queries generally leads to more performance tuning, for example, by DBAs.

```sql
SELECT c_county, c_state, calendar_year, calendar_quarter, SUM(amount_sold)
FROM store_sales, customer, date
WHERE ss_time_id=d_time_id
AND ss_cust_id=c_cust_id
AND d_calendar_month IN ('2000-09', '2000-10')
AND c_county IN ('Santa Clara', 'San Mateo')
GROUP BY
GROUPING SETS (c_county, c_state),
GROUPING SETS (calendar_year,calendar_quarter);
```

**Figure 5: Reporting Query**

#### 4.2 Add-Hoc DSS Query Class

The ad-hoc DSS query class captures the improvised nature of a DSS system in that there may not be any relationship between queries submitted to the system in sequence. In general, queries tend to be complex; however, in many cases "simple" queries are likely, for example, one may want to find out the unique attributes in a dimension table of a star schema. In the following, we give an example of a complex query in this class.

```sql
SELECT * FROM
(SELECT count_id, SUM(amount_sold), RANK() OVER (ORDER BY SUM(amount_sold) DESC) AS COUNTY_RANK
FROM store_sales, item, customer, date
WHERE ss_item_id=i.item_id
AND ss_cust_id=c_cust_id
AND ss_date_id=d_date_id
AND c_calendar_month='2000-09'
GROUP BY count_id)
WHERE COUNTY_RANK <= 5;
```

**Figure 6: Ad-hoc DSS Query**

#### 4.3 Interactive OLAP Query Query Class

While the interactive OLAP class of queries is similar to the ad-hoc DSS query class, it is distinguished by the tendency of one query to be related to the previous query. This query class models the types of queries generated by a query tool that is being used to find answers to various business questions or exception patterns. A scenario-based session consists of a sequence of queries, with one leading to another, and where the sequence may include both complex and simple queries.

```sql
WHERE COUNTY_RANK <= 5;
```

**Figure 6: Ad-hoc DSS Query**

#### 4.4 OLAP Query Class

The OLAP query class models the various types of OLAP queries. An OLAP query is a query that represents different kinds of database activities of a DSS system. The types of queries generated by a query tool that is being used to find answers to various business questions or exception patterns. A scenario-based session consists of a sequence of queries, with one leading to another, and where the sequence may include both complex and simple queries.
1. Part
create view ranking (store_id, s_name, net, rank) as
select store_id, s_name, sum(ss_net_paid) as net,
rank() over (order by sum(ss_net_paid) desc) as rank
from store_sales ss, store s, date d
where ss_s_id=s_id and d_date_id=ss_sold_date_id
and d_date between '04/01/2000' and '03/31/2001'
group by s_name;

2. Part
select * from ranking where rank < 10;

3. Part
select s_name, s_manager
sum(ss_net_paid) as sum
from store_sales, store, date
where ss_store_id=s_id
and d_date_id=ss_sold_date_id
and ss_store_id in (select s_id from ranking)
and d_date between '04/01/2000' and '03/31/2001'
group by s_name;

4. Part
select s_name, s_manager
sum(ss_net_paid) as sum
from store_sales, store, date
where ss_store_id=s_id
and d_date_id=ss_sold_date_id
and ss_store_id in (select s_id from ranking)
and d_date between '04/01/2001' and '03/31/2002'
group by s_name;

4.4 Extraction Query Class
The extraction query class typically consists of joins (no aggregates) and returns a fairly large data result set. Like the reporting queries, the query forms tend to be static.

Select c_customer_id, c_last_name,
c_first_name, c_email,
cia_zip, cd_gender, hd_dep_count
cd_marital_status, hd_vehicle_count
from customer, customer_address,
customer_demographics,
household_demographics
where c_current_addr_sk = ca_address_sk
and c_current_cdemo_sk = cd_demo_sk
and c_current_hdemo_sk = hd_demo_sk
and cd_education_status in ('BS', 'BA', 'MS')
and cd_purchase_estimate > 100
and hd_buy_potential in ('VHIGH', 'HIGH');

Figure 7: Iterative OLAP Query

5. THE DATA MAINTENANCE PROCESS
The data maintenance process models the refresh process that is commonly referred to as ETL (extract, transform, and load). In order to benchmark an ETL process that is both realistic, feasible to implement but still tests the main features of modern hardware and software for a decision support workload, TPC-DS assumes the following model.

Figure 9: Data Maintenance Process
TPC-DS models a database server centric ETL process measuring the performance of common ETL operations implemented in SQL or procedural SQL such as maintaining slowly changing dimensions, assigning surrogate keys to fact table foreign keys, data transformations and data cleansing. During a benchmark run the data warehouse is refreshed once with two percent of data obtained from flat files. That is two weeks worth of data of the database which holds 2 years of data. These collections of files, hereafter referred to as the Refresh Data Set, are structured as if they were extracted from operational systems. Hence, their structure follows the mostly 3rd normal form data of the operational systems. Furthermore, TPC-DS defines a staging area that can be used to load data from the refresh data set. Each file of the refresh data set represents the content of one table staging area. Since the refresh data set contains only partial data, the staging area does not enforce referential integrity. However, for benchmarking purposes we assume that basic data cleansing operations are already performed.

Figure 9 outlines the data maintenance process. Data provided in flat files is loaded during the first step into an optional staging area. Then SQL or programming SQL is used to transform data into the actual format that can be loaded into the target table (dimension, fact); indicated by Step 2. In the following steps (3 and 4) data is inserted into the target table. Fact table tuples are always inserted while dimension tuples are either inserted or updated depending upon their type.
5.1 Inserting Data from Data Refresh Set

The refresh data set contains new and updated data for dimensions and new data for fact tables. Data is always appended to fact tables. Fact table maintenance challenges are situated in linking foreign keys to the correct surrogate keys in each related dimension. While new dimension data is always appended to dimension tables, it depends on the Dimension type whether updated data is appended or updated. Rules for maintaining fact and dimension tables are defined in so-called Data Maintenance Functions. There are five data maintenance functions defined.

TPC-DS categorizes dimensions into four classes: Static, Combination, Historical and Non-Historical. The contents of static dimensions are loaded once during the initial database load and do not change over time. Examples of static dimensions are the time, date and income band dimensions. Combinational dimensions are similar to static dimensions as their cardinality is finite. However, the possible number of combinations is far larger than the number of combinations needed. Therefore, unlike static dimensions, combination dimensions are populated over time as new combinations occur. An example of a combination dimensions is the demographics dimension.

Historical dimensions preserve history as new dimension entries are added. Consequently, multiple dimension entries can exist for one entity. Fact table entries always point to the dimension entry that was current at the time of their load. For instance, when a customer changes residency his past purchases in the sales fact tables still refer to his previous residency, while all new sales refer to the new residency. Contrary to historical dimensions, non-historical dimensions do not keep aged data. As dimension tuples are updated, previous values are overwritten and their information is lost. All fact data is associated with the most current value of the dimension. For instance, when a customer changes his name (due to marriage) all old and new purchases refer to the same customer dimension entry (see Figure 10 for the data maintenance pseudo-code for this dimension type).

Although the exact implementation in SQL or programming SQL of the actual transformation is left to the benchmark sponsor, TPC-DS defines the transformation in SQL. This precisely defines the transformation while giving a starting point for implementations.

The following list enumerates transformations that are covered by data maintenance functions. Due to space limitations we are not able to show any of the SQL transformation.

- Type conversion (e.g. char to date)
- String manipulations (e.g. concatenation, substr, etc)
- Missing values (e.g. default values)
- Valid values (e.g. checking address extensions)
- Conditional processing (e.g. case statements)
- Mapping (e.g. female, male to 1,0)
- Un-pivoting (e.g. loading output from COBOL programs)
- Inconsistent or incorrect use of codes and special characters
- Merging with external data sources (e.g. zip codes)

6. CONCLUSION

With hard work from the member companies on the TPC-DS subcommittee, the TPC-DS benchmark is an attempt to take decision support benchmarking to the next level. Unique elements of snowflake schemas with shared dimensions, skewed data, a large query set, and a rich data maintenance component are introduced in what we hope will be a useful benchmark that will provide value to the industry in comparing different products and maybe more importantly, an arena in which products will compete to improve. It must be understood that this benchmark is still under development as of this writing, and that the aspects presented here may change before the final specification is approved. The current schedule projects this benchmark to be approved sometime in the year 2003.

7. Acknowledgments

The authors would like to thank the members of the DSS development subcommittee for their technical contributions and Sandy Venning and John Haydu for editing the paper.

8. REFERENCES