A Comparative Study of Performance Analysis and Evaluation of Oracle and DB2 on Linux

Xiuling Hu

A MAJOR REPORT

IN

THE DEPARTMENT

Of

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

August 2003

Your file Votre référence
ISBN: 0-612-83902-8
Our file Notre référence
ISBN: 0-612-83902-8

Abstract

A Comparative Study of Performance Analysis and Evaluation of Oracle and DB2 on Linux

Xiuling Hu

As much as each of us would like to get up every day with only a few things to focus on, the competitive world we live in rarely allows such a luxury. Balancing complexity with simplicity is an art, and is one that we should master. In today's competitive information technology world, the pressure is on to stay in front of your competition. The value and visibility of your development projects compel you to reduce time-to-market without sacrificing quality and with lower cost. There are so many high levels representing technologies for information system and software engineering. With so many representing technologies, how do we choose the one that will help us get ahead at the front line? The purpose of this project is to study the capabilities of Oracle & DB2 in the Linux Operating System, and to compare the two according to TPC-H benchmark. We consider system performance, price, and user interfaces.

Acknowledgements

I would like to thank sincerely my supervisors, Dr. Shiri and Dr. Grahne, for their guidance, support, encouragement and friendship. Whenever I need help, Dr. Shiri is always there. Thanks for his patience and kindness. Furthermore, thanks the system analysts of CS Department of Concordia University for creating the experimental environment for this report.

I am grateful to professors and staffs in Computer Science department at Concordia University for the wonderful courses and services. I would especially like to say thanks to Ms. Halina Monkiewicz, who was always friendly and prompt in providing assistance.

Finally, I would like to express my deep gratitude to my husband, George Wang, for his support.

Table of Contents

List of Figures

1. Introduction

1.1 Problem Statement

Every database vendor claims, its database is fast. Consider the following two sample announcements from IBM and Oracle:

IBM announces DB2 is the most scalable database in the world, combined with the most scalable Unix servers reaches a new milestone in enterprise data warehousing. DB2 runs the leading 10TB warehouse benchmark on IBM's fastest computer. It has been in the lead across all scale factors for TPC-H (100GB, 300GB, 1TB, 3TB and 10TB), DB2 has demonstrated sustained leadership in TPC-H over the life of the benchmark [1].

Oracle announces a record-breaking data warehousing benchmark for Oracle9i Database Release 2 on the HP 9000 Superdome Enterprise Server. Delivering higher performance with half the number of processors, this new world record TPC-H 3TB result outperforms IBM DB2's best result by 29%, and costs 24% less per QphH@3000GB. Oracle9i on HP also surpasses NCR Teradata's best 3TB result by 44% with half the number of processors and costs 78% less per QphH@3000GB [2].

QphH@3000GB presents the performance metric of a 3000GB database. The detail information about definition and formula is on page 11.

As we can see above, they have used different hardware platforms, different CPUs, memory and disk drives. In the IBM's test above, the memory size is 256GB. In reality, we do not yet dream to work with such a huge memory. So it is difficult to compare benchmark results for these products because of such differences. The question is: "As database users, how do we know which DBMS is really better or worse? " The objective of this project is to provide some answers to this question.

1.2 Scope and Organization of This Report

In this report, we present our TPC-H benchmark experiments for 1GB and 10GB data in Oracle 9.0.1 and DB2 7.2, both on Red Hat Linux 7.3 environment. We report our experiments and observations.

Possible tests on Linux are listed in Figure 1.

Size / DBMS	1GB	10GB
Oracle	test1	test3
DB2	test2	test4

Figure 1. Tests List

Test1 is about 1GB data in Oracle, test2 is about 1GB data in DB2, test3 is about 10GB data in Oracle, and test4 is about 10GB data in DB2. We compare test1 with test2 and test3 with test4. The comparison will be based on not only the performance, but also include the price and user interface developed. With the help of this comparison, we will be able to get an idea about which DBMS is better for a given data size and OS type. This study also provides a guideline in order to choose a DBMS and OS that is suitable for a particular application size and environment.

A similar study on Windows 2000 platform was done in parallel [4]. In section 4, we will compare the results in [4] with ours on Linux.

In order to compare between Oracle and DB2, same hardware was used in this experiment. We used DELL PC with one 1.7 GHZ CPU, 256 MB RAM and two 40GB hard disks; We even used the same system setup, for instance, the swap space was 500 KB, the Linux kernel parameter shmmax was 33,554,432 B; We used the same TPC-H parameters, for example, we used the same seeds to generate data and queries, so that the data and queries were the same for Oracle and DB2. Some concepts of Oracle and DB2 are similar, some of them are different. For instance, block size in Oracle is called page size in DB2. Both Oracle and DB2 have system and temporary table spaces. Oracle has undo table space, but DB2 does not. Due to these kinds of difference, it is not possible for these two DBMS to have exactly the same setup. However, we could allocate same amount of total table spaces for both DBMS, such as 3.3GB for 1G data test, and 35GB for 10GB data test. We used the default values for all other DBMS parameters setup. We assume the default values of DBMS parameters are the best for general cases. Also, in order to avoid caching data and SQL in RAM, we reboot the computer before starting a test.

The rest of this report is organized as follows: In section 2, we give an overview of TPC benchmark with special emphasis on TPC-H. In section 3, we present our experiments of TPC-H benchmark test for 1GB and 10GB data in Oracle 9.0.1 and DB2 7.2 on Red Hat

Linux 7.3, the results, metrics, problems and comments. In section 4, we compare the performance between Oracle and DB2 on Linux. In addition to this, we compare the Oracle/DB2 performance on Linux and Window 2000. Performance tuning is discussed in section 5. Finally, section 6 provides the concluding remarks.

2. A Review of TPC-H Benchmark

In this section, we review TPC benchmark in general, and TPC-H in particular. These introduction materials are borrowed from [3].

The TPC (Transaction Processing Performance Council) is a non-profit corporation founded to define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry. The TPC recognizes that different types of applications have different processing requirements. The TPC-C benchmark is developed for online transaction processing (OLTP) benchmarks, and attempts to simulate real world OLTP database applications. It is in fact the de facto standard for OLTP. The TPC-W benchmark for Web-based systems. The TPC-R and TPC-H (formerly TPC-DS) benchmarks are developed for data warehouses and decision support systems.

The TPC Benchmark™H (TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance. This benchmark illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions [3]. The TPC-H consists of a set of business queries designed to exercise system functionalities in a manner representative of complex business analysis applications. These queries have been given a realistic context, portraying the activity of a wholesale supplier to help the reader relate intuitively to the components of the benchmark [3]. TPC-H does not represent the activity of any particular business segment, but rather any industry which must manage, sell, or distribute a product worldwide (e.g., car rental, food distribution, parts, suppliers, etc.). TPC-H does not attempt to be a model of how to build an actual information analysis application.

2.1 Database Characteristics

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 is able to support queries and refresh functions against all tables on a 7 day by 24 hour (7 x 24) basis.

The component of the TPC-H database consists of eight base tables. The schemas of these tables and the relationships between these tables are illustrated in Figure2.

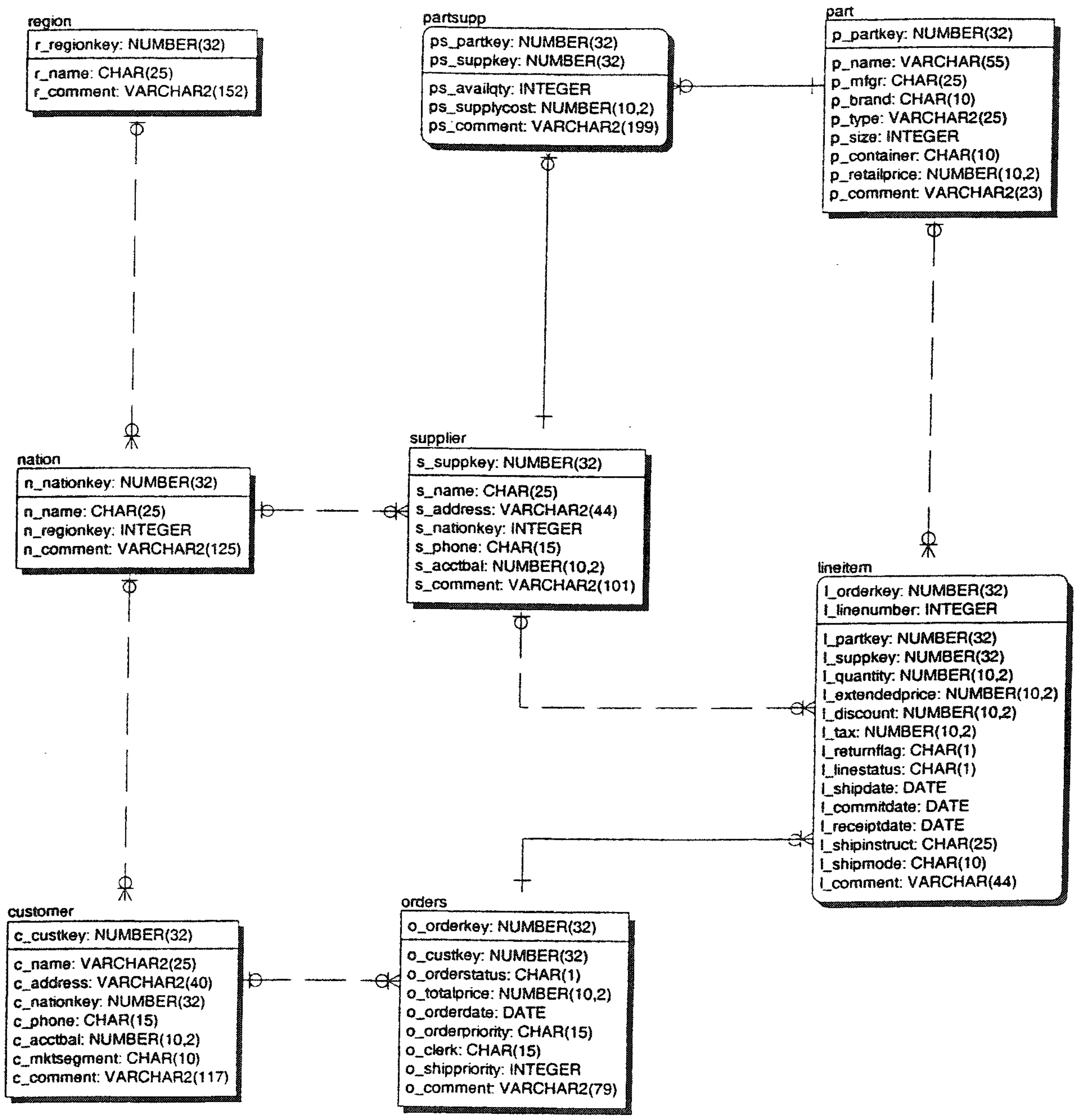


Figure 2. TPC-H Schema

2.2 Queries and Refresh Functions

The purpose of TPC-H benchmark is to reduce the diversity of operations found in an information analysis application, while retaining the application's essential performance characteristics [3]. There are twenty-two different queries which have a high degree of complexity. Appendices illustrate these queries. Many of the queries are not of primary interest for performance analysis because of the length of time the queries run, the system resources they use, and the frequency of their execution [3]. The queries considered in this benchmark exhibit the following characteristics:

- They have a high degree of complexity;
- They use various access methods;
- They are ad hoc in nature;
- They examine a large percentage of the available data;
- They all differ from each other;
- They contain query parameters that change across query executions.

While the benchmark models a business environment in which refresh functions are an integral part of data maintenance, the refresh functions used in the benchmark do not attempt to model this aspect of the business environment [3]. The refresh functions demonstrate update functionality for the DBMS, while assessing an appropriate performance cost to the maintenance of the auxiliary data structure. There are two refresh functions (RF) used in this benchmark as follows:

- RF1: add sales information to the DB
- RF2: remove sales information from the DB

2.3 Data and Queries Generation Programs

QGEN and DBGEN are C programs, which can be downloaded from TCP-H web page: http://www.tpc.org/tpch/default.asp. QGEN program is used to generate the executable query text and DBGEN program is used to generate the data. According to the concrete syntax of the database system under test, these queries should be modified to run on the specific database. However, modifications are limited for only syntax matching. Any other modifications such as speeding up etc. are not allowed. A parameter seed to the random number is used to generate substitute parameters. The selection of seed should follow the rules below:

- An initial seed (seed0) is first selected as the time stamp of the end of the database load time expressed in the format *mmddhhmmss*, where *mm* is the month, *dd* is the day, *hh* is the hour, *mm* is the minutes, and *ss* is the seconds. This seed is used to seed the power test of Run1.

- Other seeds (for the throughput test are chosen as seed0 + 1, seed0 + 2, ..., seed0 + n, where n is the number of throughput streams selected by the vendor.

- Sponsor decides whether Run2 should use the same seeds as the Run1, but method of selecting seeds should be the same.

Run1 and Run2: a performance test consists of two runs. The detail information is in section 2.6 on page 11

2.4 Database Load

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

The total elapsed time to prepare the test database for the execution of the performance test is called the database load time. This includes the elapsed time to create the tables, load data, create indices, define and validate constraints, and gather statistics for the test database.

The population of the test database consists of two logical phases:

- Generation: the process of using DBGEN to create data in a format suitable for presentation to the DBMS load facility.
- Loading: the process of storing the generated data into the database tables.

2.5 Power Test and Throughput Test

A power test is to measure the raw query execution power of the system when connected with a single active user. In this test, a single pair of refresh functions are executed exclusively by a separate refresh stream and scheduled before and after execution of the queries [3]. The power test consists of three execution streams in order: refresh function1 stream, power test queries stream, and refresh function2 stream. Timing intervals for each query and for both refresh functions are collected and reported for the performance calculation.

The sequence of queries in a power test is as follows:

Stream0: 14, 2, 9, 20, 6, 17, 18, 8, 21, 13, 3, 22, 16, 4, 11, 15, 1, 10, 19, 5, 7, 12.

A throughput test is to measure the ability of the system to process most queries in the least amount of time [3]. The throughput test is where test sponsors can demonstrate the performance of their systems against a multi-user workload. A throughput test consists of S number of query streams, remaining constant during the whole measurement interval. In addition, another refresh stream should be parallel to those S streams. The S is an integer number, and is decided by the size of the testing database. When the database size is 1GB, S is 2. When the database size is 10GB, S is 3.

The sequence of queries in a throughput test as follows:

Stream1: 21, 3, 18, 5, 11, 7, 6, 20, 17, 12, 16, 15, 13, 10, 2, 8, 14, 19, 9, 22, 1, 4.

Stream2: 6, 17, 14, 16, 19, 10, 9, 2, 15, 8, 5, 22, 12, 7, 13, 18, 1, 4, 20, 3, 11, 21.

Stream3: 8, 5, 4, 6, 17, 7, 1, 18, 22, 14, 9, 10, 15, 11, 20, 2, 21, 19, 13, 16, 12, 3.

2.6 Execution Rules

The benchmark is defined as the execution of the load test followed by the performance test. The load test begins with the creation of the database tables and includes all activity required to bring the system under test to the configuration that immediately precedes the beginning of the performance test. The load test does not include the execution of any queries in the performance test or similar query.

The performance test consists of two runs. A run consists of one execution of the Power test and one execution of the Throughput test. Throughput test must follow, one and only one, power test. No activity that improves the system performance is allowed between the power test and the throughput test. Run1 follows the data load and Run2 follows Run1. If Run1 fails, the benchmark must be restarted with a new load test. If Run2 fails, it may be restarted without a reload.

All sessions supporting the execution of a query stream have been initialized in exactly the same way. The initialization of the session supporting the execution of the refresh stream may be different than that of the query streams.

2.7 Performance Metric

The performance metric reported by TPC-H is called the TPC-H Composite Query-per-Hour Performance Metric (QphH@Size). It reflects multiple aspects of the capability of the system to process queries, including the selected database size against which the queries are executed, the query processing power when queries are submitted by a single stream, and the query throughput when queries are submitted by multiple concurrent users. The TPC-H Price/Performance metric is expressed as Price($) /QphH@Size.

- The TPC-H Composite Query-per-Hour Performance Metric

  The numerical quantities of "power" and "throughput" are combined as follows to form the TPC-H composite query-per-hour performance metric:

$$\text{QphH@size} = \sqrt{\text{Power@size} * \text{Throughput@size}}$$

The unit of QphH@Size is Queries per hour* Scale-Factor, reported to one digit after the decimal point.

- The TPC-H Price/Performance Metric

The TPC-H Price/performance metric at the chosen database size, TPC-H Price-per-QphH@Size, is computed using the performance metric QphH@Size as follows:

TPC-H Price-per-QphH@Size = Price ($) / QphH@size

Where:

$ is the total system price, in the reported currency.

QphH@Size is the composite query-per-hour performance metric.

Size is the database size chosen for the measurement.

Furthermore, the formulas for calculating power metric and throughput metric are as follows.

$$\text{TPC-H Power@Size} = (3600 * SF) / \sqrt[24]{\left(\prod_{i=1}^{i=22} QI(i,0) * \prod_{j=1}^{j=2} RI(j,0)\right)}$$

$$\text{TPC-H Througthput@Size} = (S * 22 * 3600) * SF / T_s$$

Where:

Size is the database size chosen for the measurement and SF is corresponding scale factor. For example, if Size=10GB, then SF=10.

S is the number of query streams used in throughout test.

QI(i,0) is the timing interval, in seconds, of query Q_i within the single query stream of the power test

Ts represents the execution time of the throughput test. It is defined as follows:

- It starts either when the first character of the executable query text of the first query of the first query stream is submitted by the driver, or when the first character requesting the execution of the first refresh function is submitted by the driver, whichever happens first.
- It ends either when the last character of output data from the last query of the last query stream is received by the driver, or when the last transaction of the last refresh function has been completely and successfully committed and a success message has been received by the driver, whichever happens last.

The TPC-H performance test consists of two runs: Run1 and Run2. The reported performance metric must be for the run with the lower TPC-H Composite Query-Per-Hour Performance Metric because the TPC-H metrics reported for a given system must represent a conservative evaluation of the system's level of performance. Each of these includes a power test and a throughput test. The former measures the raw query execution power of the system when connected with a single active user, whereas the later measures the ability of the system to process most of queries in the least amount of time.

3. Experiments

In our experiment of TPC-H benchmark, we use Oracle 9.0.1 and DB2 7.2 on Red Hat Linux 7.3 on DELL PC with one 1.7 GHZ CPU, 256 MB RAM and two 40GB hard disks. There are four tests. Test1 is with 1GB data in Oracle, test2 is with 1GB data in DB2, test3 is with 10GB data in Oracle, and test4 is with 10GB data in DB2.

3.1 Preparation of Experiment

Before we start a test, we need to install Oracle and DB2 software, create the database schemas, and prepare data that will populate the databases and queries that will be used to evaluate the performance.

3.1.1 Installation of software

The installation of Oracle and DB2 on Linux is more difficult than on Windows. We have experienced some difficulties during the installation.

3.1.1.1 Installing Oracle

Oracle provides Oracle9i Installation Guide Release 1 (9.0.1) for UNIX Systems: AIX-Based Systems, Compaq Tru64 UNIX, HP 9000 Series HP-UX, Linux Intel and Sun SPARC Solaris, which supports many platforms. After reading the whole document, we still do not understand how to install Oracle on Linux. For instance, we do not know whether JDK should be installed separately, or it is installed automatically during Oracle installation. Also the document asks user to check swap space, shared memory, but does not say how. We found a useful article on internet [5], this article provides an step-by-step installation guide of Oracle 9i. For example, it describes how to check swap space, shared memory, development packages and disk space, how to add temporary swap space, how to temporarily increase shared memory, how to install JDK, and so on. It also includes a list of Oracle 9i (9.0.1 & 9.2.0) installation problems that the author has experienced or have been posted by others. For some reason, we had to install Oracle 9.0.1 3 times, we experienced the different problems every time, most problems were listed in that article, and we got the solutions there. In our third attempt to install Oracle, the "Oracle Net Configuration Assistant" hung for 5 hours when runInstaller started to configure the tools ("Configuration Tools"). For that, we had to stop the Configuration Assistant, and manually launched Net Configuration Assistant (NETCA) and Database Configuration Assistant (DBCA).

3.1.1.2 Installing DB2

It was not quite clear from the document “Installing and Configuring DB2 Universal Database on Linux” how to install DB2 Universal Database Enterprise Edition Version7.2 for Linux. Luckily, we found [6] to be a useful guide for the installation procedure.

After we installed DB2 successfully, we got errors when we tried to launch DB2 control center (db2cc), because our *localhost* and *hostname* were different. DB2 on Linux requires these two names to be identical. This, however, was not found in the formal documents. After we changed *localhost* and *hostname* to the same and re-installed DB2, we were able to start db2cc successfully.

Compared to Oracle, installation of DB2 is easier and faster with less problems. But the user interface of installation of Oracle is much better than DB2. The corresponding user interface of DB2 looks as if the software was developed 2 decade ago.

3.1.2 Creation of Database

After Installation of Oracle and DB2 software, we are not ready for experiments yet. Before we start a test, we need to create databases, tables and indices, and generate the queries and data.

A database is a collection of data, consists of one or more data files. A data file consists of one or more table spaces. A table space is a set of segments. A segment is a set of extents. An extent is a number of contiguous data blocks. A data block is the smallest unit of I/O used by a database. This data block size should be a multiple of the operating system's block size which is 4KB in Linux.

During creating a database, we can customize initial parameters or simply accept their default values. Most parameter can be changed after the database is created. But, there are a few parameters which cannot be changed once the database is created, such as data block size.

In our TPC-H benchmark experiment, we used the default values for all parameters setup. For instance, the default data block size is 8KB in Oracle, and the default page size is 4KB in DB2. Besides the benchmark experiments, we also did other experiments for performance tuning purpose. For instance, we did some experiments to see how data block sizes, statistics information, undo table space, and temporary table space affect database performance. Collecting or deleting statistics, changing the size of undo table space or temporary table space can be done after the database is created. But the block size can not be changed once the database is created.

3.1.2.1 Creating the Database in Oracle

In Oracle, we created three databases. Their data block sizes were 4KB, 8KB, and 16KB respectively. Other initial parameters were the same. 8KB-block-size database was for the TPC-H benchmark experiment, others were for performance tuning.

We have two options for creating our Oracle database: use the Oracle Database Configuration Assistant (DBCA) which is a graphical user interface (GUI) tool, or create the database manually from a script which is based on an existing database or a sample Oracle provides.

- Database with 8KB data block size

We used DBCA to create the database for our TPC-H benchmark experiment. All initial parameters were the default values that DBCA provided, such as data block size is 8KB by default. After the database was created, we changed the size of table spaces to hold 1GB or 10GB data.

- Database with 4KB and 16KB data block size

In order to see how block size inferences database performance, we need to create the databases with 4KB and 16KB data block sizes. At first, we tried to use DBCA to create them, we got error message "ORA-00058: DB_BLOCK_SIZE must be 8192 to mount this database". We generated a script based on 8KB-data-block database, and changed the block size from 8KB to 4KB and 16KB. Then we created the databases using SQL* Plus manually.

3.1.2.2 Creating the Database in DB2

In DB2, data block is called page. DB2 Control Center utility provides graphic user interface to create databases. There are three kinds of table spaces: system, user, and temporary table spaces in a database.

- Database with 4KB page size

The initial page size is 4KB by default in DB2. One system table space, one temporary table space, and one user table space were created by default when we created the database using DB2 Control Center utility. We used this database with all default initial values as our TPC-H benchmark database.

- Database with 8KB page size

In order to see how block size inferences the performance, we need to have a database with 8KB block size to compare the performance with 4KB-block-size database. There is no way to create a system or temporary table space with 8KB block size in DB2. However, we can create a user table space with 8KB page size and drop the original 4k-page-size user table space. In this case, the page sizes of system and temporary table spaces have to be 4k. So, in our 8KB-block-size database in DB2, actually only the user table space has 8KB page size. The page sizes of system and temporary table spaces were still 4KB.

3.1.3 Generation of Data

DBGEN, which is provided by TPC, is used to generate qualified data. However, the original standard C program did not work on Linux; there were compiling errors. We had to modify this program so that it could run on Linux. The modified program is in the enclosed CD.

In order to generate 1GB data for all tables of TPC-H database, the command "*dbgen –s 1*" was used. In order to generate 10GB data for all tables of TPC-H database, the command "*dbgen –s 10*" is supposed to be used. In doing so, we got error message "*File size limit exceeded*". This is because Red Hat Linux 7.3 has a bug that limits file size to 2G. However, the size of table *lineitems* is about 7GB. We had to use additional parameters to generate data for every table individually and generate 5 pieces for the biggest table *lineitem* to avoid 2G limit. The following commands are used to achieve this:

dbgen –s 10 –T c	to generate data for table *customer*
dbgen –s 10 –T s	to generate data for table *supplier*
dbgen –s 10 –T n	to generate data for table *nation*
dbgen –s 10 –T r	to generate data for table *region*
dbgen –s 10 –T O	to generate data for table *order*
dbgen –s 10 –T P	to generate data for table *part*
dbgen –s 10 –T S	to generate data for table *partsup*
dbgen –s 10 –S 1 –C 5 –T L	to generate the first part for table *lineitem*

dbgen –s 10 –S 2 –C 5 –T L to generate the second part for table *lineitem*

dbgen –s 10 –S 2 –C 5 –T L to generate the third part for table *lineitem*

dbgen –s 10 –S 2 –C 5 –T L to generate the forth part for the table *lineitem*

dbgen –s 10 –S 2 –C 5 –T L to generate the fifth part for table *lineitem*

Five data files for table *lineitem* have been generated as above. These data files contain raw data which will be populated to the databases during load tests. There is no difference between one raw data file and five raw data files once all data is populated to databases. It does not affect databases performance at all.

3.1.4 Generation of Queries

3.1.4.1 Generating Queries and Sequences

The 22 queries used in the experiment are generated by QGEN, provided by the TPC-H. Since there were compiling errors, we had to modify this program so that it could run on Linux. The modified program is in the enclosed CD.

As discussed in Section 2.2.3, we used the seed to generate the random number for the parameters substituted in the original queries.

The required minimum number of query streams for throughput is 2 for 1GB size and 3 for 10GB. For 1GB data, stream1 and stream2 are used for run1 and run2, whereas for 10Gdata, stream1, stream2, and stream3 are used for run1 and run2. In our project, run1 and run2 use different seeds. All seeds used to generate query sequences are shown as Figure 3:

Seed / Run — Stream	Run1	Run2
Stream0(power)	807140330	807210122
Stream1(throughput)	807140331	807210123
Stream2(throughput)	807140332	807210124
Stream3(throughput)	807140333	807210125

Figure 3. Seeds for streams

3.1.4.2 Generating Refresh Functions

A Refresh function includes a sequence of Insert refresh function (RF1) and Delete refresh function (RF2).

DBGEN can be used to generate raw data for RF1 and RF2 using specified parameters. Furthermore, the parameter –S n should be given for the specified database scale factor. For example, for the 1GB database, we used the following command line:

```
dbgen –S 1 –U6
```

For the 10GB database, we used the following command line:

```
dbgen –S 10 –U8
```

The result of each of the above command line is the raw data, instead of SQL statements. In order to get the refresh functions with SQL statement, we developed two programs using *Perl* to generate the sequence of RF1 and RF2 automatically. The *Perl* programs (rf1.pl and rf2.pl) are in the enclosed CD.

3.1.4.3 Validating Queries

After having generated the queries, query validation must be done. Query validation describes how to validate the query against qualification database. Each query has one or more substitution parameters. When generating executable query text, a value is supplied for each substitution parameter of that query. These values are used to complete the executable query text. These substitution parameters are expressed as names in uppercase and enclosed in square brackets. According to the concrete syntax of Oracle and DB2, some queries had to be modified to run on the specific database. However, we did only modification to match syntax, we did not do any other modification which whould speed up the process.

The functional query definition uses the following minor modification on each DBMS respectively:

Oracle

- Date fields use Oracle date function. For example, to_date(date '1998-03-21')
- The standard Oracle date syntax is used for the date arithmetic. For example, to_date(date '1996-02-21' + interval '5' days)
- Queries 2, 3, 10, 13, and 21 were modified in order to fetch the given number of the query result. the *rownum* < *n* is used in the *where* clause of those queries, where *n* is an integer, denoting the number of rows in the query output.

DB2

- The standard IBM date syntax is used for the date arithmetic. For example, date ('1996-02-21')+5 days

- Queries 2, 3, 10, 13, and 21 were modified in order to fetch the given number of the query result. The "*fetch last n rows*" is used in *where* clause of those queries, where n is an integer, denoting the number of rows in the query output.

3.2 Utilities

There are many utilities in Oracle, DB2 and Linux, some of which are used to carry out our experiment and make it easier. Below is the list of the utilities we use in this experiment.

3.2.1 Linux Utilities

- crontab

  A *cron* is a utility that allows the tasks to run automatically in the background at regular intervals by use of the *cron* daemon. *Crontab* is a file which contains the schedule of *cron* entries to be run and at what times they are to be run. This was quite useful in our experiment. For example, in Oracle run1 test with 10G data, loading data and updating statistics took about 4 hours, refresh1 of power test took about 10 minutes, executing 22 queries took about 3 hours, refresh2 took about 10 minutes; whole test took nearly 12 hours. Because of *crontab* utility, we did not have to stay in the lab and wait for the whole lengthy period of experiment. We used *crontab* to schedule the tasks to run automatically, after completing run1 test, We rebooted the computer and change *crontab* for run2 test.

3.2.2 Oracle Utilities

- runInstaller

  To install Oracle.

- Net Configuration Assistant (NETCA)

  To setup Net*8 connection.

- Database Configuration Assistant (DBCA)

  To create databases.

- Oracle Enterprise Manager to create data

  To manage databases.

- SQL Plus

  To execute SQL commands.

3.2.3 DB2 Utilities

- db2setup

  To install DB2.

- db2sart & db2stop

  To start & stop a DB2 instance.

- db2admin start & db2admin stop

  To start & stop the DB2 Administrative Server.

- dbjstrt

To start the JDBC listener process.

- db2cc &6789

  To launch db2 control center using the default port 6789.

- db2batch

  To launch batch file for executing DB2 commands.

From DB2 commands above, we can see that there is no space between *db2* and *start* in *db2start*; there is a space between *db2admin* and *start* in *db2admin start*; there is no space and using *strt* instead of *start* in *db2jstrt*. These commands do not meet the same name convention.

3.3 Test

A test consists of two parts: load test and performance test. The load test contains database schema creation, data load, and statistics collection. The performance test consists of two runs. A run consists of one execution of the Power test followed by execution of the Throughput test. We reboot the computer every time before we start a test to make sure there is no caching data and SQL in RAM. This is to ensure that all tests have exactly the same initial status. The procedures are as follows:

- Reboot the computer
- Load test
  - Create database schemas
  - Load data and collect statistics
- Performance test
  - Run1
    - Power test (refresh11, query stream1, refresh12 serially)

        - Throughput test (refresh1, stream11, stream12, stream13 in parallel)
    - Reboot the computer
    - Run2
        - Power test (refresh21, query stream2, refresh22 serially)
        - Throughput test (refresh2, stream21, stream22, stream23 in parallel)

3.3.1 Load Test

The total elapsed time to prepare the test database for the execution of the performance test is called *database load time*. Since the elapsed time to create the tables and indices, define and validate constraints is less than 1 second, it is ignored in our experiments. The database load time only includes the elapsed time to load data and gather statistics for the test database. The database load time is given below:

block size \ seconds \ Database	Oracle	DB2
4KB	1545	1386
8KB	1252	1316

Figure 4. Database load time for 1GB data

Database	Oracle	DB2
Load Time(seconds)	10841	11380

Figure 5. Database load time for 10GB data

As mentioned earlier, the default block size is 8KB in Oracle and 4KB in DB2. From the above Figure 4, we can see that in 1GB load test, Oracle is faster than DB2 with 8KB block size. However, DB2 is faster than Oracle with 4KB block size. From the above Figure 5, we can see that the load test of Oracle is faster than that of DB2 for 10GB data size. In 10GB test, we used the default block size, which is 8KB in Oracle, and 4KB in DB2. In addition, all time intervals were recorded by system automatically in Oracle because Oracle provides commands to count time for loading data and gathering statistics. In DB2, the time intervals for loading data were recorded by the system automatically, however, the time intervals for gathering statistics were recorded manually. We used the computer clock, and noticed the time when the gathering statistics process started and stopped. It was not an easy task, and the results were not quite accurate.

3.3.2 Performance Test

As mentioned earlier, a TPC-H performance test consists of two runs: Run1 and Run2. A run consists of one execution of the Power test and one execution of the Throughput test. The power test contains a pair of refresh function and 22 queries which run serially. The execution time of these functions and queries are recorded for the purpose of calculating TPC-H power metric. The throughput test follows the power test. There is no rebooting action between the power test and the throughput test. A throughput test contains a set of query streams and a refresh stream which run concurrently. For 1GB size database, the number of streams in throughput test is 2, and for 10GB size database, this number is 3.

The run with the lower TPC-H Composite Query-Per-Hour Performance Metric should be adopted as reported performance metric. The price is also a contribution to calculate performance metrics. The price of the computer system that we use to do experiment is about $4,000. The price of Oracle 9i is about $63,348, and the price of DB2 7.2 is about $43,686. So the total price in Oracle test is $67,348, and the total price in DB2 test is $47,686.

3.3.2.1 Test1 –1GB Size In Oracle

Test1 involves 1GB data in Oracle. In this test, the user table space is 1600MB, the system table space is 500MB, the undo table space is 20MB, the temporary table space is 1000MB, and the size of total table spaces is 2.3GB. All other database parameters used are the default values. For instance, the value of db_block_size is 8KB by default. The following Figure 6 shows the execution timing intervals, in seconds, which are used to calculate the Performance Metrics.

query	run1 (seconds)				run2 (seconds)			
	power1	throughput1			power2	thoughtput2		
		stream11	stream12	refresh1		stream21	stream22	refresh2
Q01	48	56	79		48	113	64	
Q02	16	9	38		20	24	88	
Q03	9	18	8		7	16	49	
Q04	30	35	218		30	56	30	
Q05	8	111	87		71	125	57	
Q06	26	65	50		20	36	43	
Q07	55	124	74		53	120	386	
Q08	34	40	34		29	57	29	
Q09	53	165	100		92	213	120	
Q10	32	56	68		31	82	74	
Q11	7	17	7		7	20	198	
Q12	27	336	276		26	79	25	
Q13	28	68	82		26	125	61	
Q14	34	28	44		45	39	205	
Q15	46	54	76		48	90	97	
Q16	11	24	26		7	56	73	
Q17	45	61	78		41	121	90	
Q18	34	118	115		32	145	43	
q19	27	45	80		29	59	39	
q20	33	55	29		26	38	16	
q21	33	287	98		94	234	12	
q22	9	12	12		9	25	149	
rf1	46				25			
rf2	17				17			
sum	708	1784	1679	795	833	1873	1948	1419

Figure 6. Result of test1

Where:

the first column is the query sequence, rf1 and rf2 are the refresh functions used in the power test,

power1 is the power test of run1,

stream11 and stream12 are the query streams of run1,

refresh1 is the refresh function of run1,

power2 is the test of run2,

stream21 and stream22 are the query streams of run2,

refresh2 is the refresh function of run1, sum shows the summary for each column,

From Figure 6, we can see: in run1, power test took 708 seconds; throughput test took 1784 seconds in stream1, and 1679 seconds in stream2; refresh function took 795 seconds. In run2, power test took 833 seconds; throughput took 1873 seconds in stream1, and 1948 seconds in stream2; refresh function took 1419 seconds.

Performance Metrics

Using the formulas described in section 2.2.7, we can get the following performance metrics:

S=2, SF=1, Price=$ 67348, Ts=1784 in run1, Ts=1873 in run2.

Run1

TPC-H Power@1GB = 143.3 (query-per-hour)
TPC-H Throughput@1GB = 88.69 (query-per-hour)
QphH@1GB = 112.7 (query-per-hour)
TPC-H Price-per-QphH@1GB = Price ($) / QphH@1GB = 597.27 $

Run2

TPC-H Power@1GB = 131.38 (query-per-hour)
TPC-H Throughput@1GB = 81.31 (query-per-hour)
QphH@1GB = 103.3 (query-per-hour)
TPC-H Price-per-QphH@1GB = Price ($) / QphH@1GB = 651.59 $

The QphH@1GB in Run2 is less than that in Run1. According to section 2.7, the result of Run2 is considered as the final result of 1GB DB2 test as Figure 7.

QphH@1GB	103.3 query-per-hour
TPC-H Price-per-QphH@1GB	651.59 $

Figure 7. Metrics of test1

3.3.2.2 Test2 –1GB Size In DB2

Test2 involves 1GB data in DB2. In this test, the user tables pace is 1600MB, the system table space is 500MB, the temporary table space is 200MB, and the size of total table spaces is 2.3GB as well. All other database parameters used are the default values. For instance, the page size is 4KB by default. The following Figure 8 shows the execution timing intervals, in seconds, which are used to calculate the Performance Metrics.

query	run1 (seconds)				run2 (seconds)			
	power1	throughput1			power2	thoughtput2		
		stream11	stream12	refresh1		stream21	stream22	refresh2
q01	112	113	242		115	129	243	
q02	6	28	231		6	35	222	
q03	90	550	159		91	558	157	
q04	52	10273	893		52	8424	943	
q05	71	142	105		72	4462	188	
q06	74	47	304		33	51	296	
q07	82	115	118		80	118	182	
q08	86	167	160		86	173	145	
q09	1009	1135	10298		990	1129	9445	
q10	66	166	219		69	131	185	
q11	38	57	64		39	55	64	
q12	50	127	97		58	80	79	
q13	78	143	190		78	119	127	
q14	65	147	148		50	87	130	
q15	55	107	3271		53	48	3133	
q16	9	25	23		9	30	17	
q17	4905	6797	5062		4316	6474	5162	
q18	77	4706	129		44	286	80	
q19	61	55	1254		61	182	135	
q20	8798	7087	10372		7063	6492	8448	
q21	187	190	195		181	200	280	
q22	60	141	175		60	104	89	
rf1	90				87			
rf2	37				32			
sum	16158	32318	33709	32906	13725	29367	29750	30103

Figure 8. Result of test2

Where:

the first column is the query sequence, rf1 and rf2 are the refresh functions used in the power test.

power1 is the power test of run1.

stream11 and stream12 are the query streams of run1.

refresh1 is the refresh function of run1.

power2 is the power test of run2.

stream21 and stream22 are the query streams of run2

refresh2 is the refresh function of run1

sum shows the summary for each column.

From Figure 8, we can see: in run1, power test took 16158 seconds; throughput test took 32318 seconds in stream1, and 33709 seconds in stream2; refresh function took 32906 seconds. In run2, power test took 13725 seconds; throughput took 29367 seconds in stream1, and 29750 seconds in stream2; refresh function took 30103 seconds.

Performance Metrics

As in the test1 case, we use the formulas described in section 2.2.7 to get the performance metrics as follows:

S=2, SF=1, Price=$ 47686, Ts=33709 in run1, Ts=30103 in run2.

Run1

TPC-H Power@1GB = 37.96 (query-per-hour)
TPC-H Throughput@1GB = 4.7 (query-per-hour)
QphH@1GB = 13.3 (query-per-hour)
TPC-H Price-per-QphH@1GB = Price ($) / QphH@1GB = 3569.3 $

Run2

TPC-H Power@1GB = 41.26 (query-per-hour)
TPC-H Throughput@1GB = = 5.26 (query-per-hour)
QphH@1GB = 14.7 (query-per-hour)
TPC-H Price-per-QphH@1GB = Price ($) / QphH@1GB = 3237.34 $

The QphH@1GB in Run1 is less than that in Run2. According to section 2.7, the result of Run1 is considered as the final result of 1GB DB2 test as Figure 9.

QphH@1GB	13.3 query-per-hour
TPC-H Price-per-QphH@1GB	3569.3 $

Figure 9. Metrics of test2

3.3.2.3 Test3 –10GB Size In Oracle

Test3 involves 10GB data in Oracle. In this test, user table space is 16GB, system table space is 1GB, undo table space is 4GB, temporary table space is 14GB, size of total table spaces is 35GB. All other database parameters used are the default ones. For instance, the value of db_block_size is 8KB by default. The following Figure 10 shows the execution timing intervals, in seconds, which are used to calculate the Performance Metrics.

query	run1(seconds)					run2(seconds)				
	power1	throughput1				power2	thoughtput2			
		stream11	stream12	stream13	refresh1		stream21	stream22	stream23	refresh2
q01	472	853	1741	867		470	809	1782	1333	
q02	134	342	911	462		138	753	890	810	
q03	688	3609	1883	739		641	4432	2437	1012	
q04	380	1857	4763	1960		384	978	918	1473	
q05	680	1518	1699	2591		666	2694	2766	2862	
q06	248	632	907	547		267	1031	793	1480	
q07	672	2202	1507	1962		645	3608	3663	3110	
q08	370	1100	652	1653		369	1467	1326	1961	
q09	1594	8008	6553	8306		1598	9078	11847	12279	
q10	521	1368	1586	1919		521	3345	3130	2428	
q11	75	341	151	107		75	466	270	300	
q12	496	4977	6176	6753		510	1120	1613	631	
q13	1012	4593	2768	4877		1090	6706	7490	3993	
q14	297	413	680	565		321	999	1021	981	
q15	542	1041	1641	854		543	2269	1893	2272	
q16	169	576	509	755		187	1022	858	322	
q17	519	1472	1668	1267		523	1726	1612	2297	
q18	338	1047	1357	1022		339	2627	2067	2080	
q19	784	1950	1305	1798		751	3120	2698	2620	
q20	313	628	667	407		318	1023	719	896	
q21	1381	4342	3877	4311		1390	7486	3968	10465	
q22	125	257	214	249		133	251	403	316	
rf1	210					213				
rf2	169					177				
sum	12189	43126	43215	43971	2046	12269	57010	54164	55921	2120

Figure 10. Result of test3

Where:

the first column is the query sequence, rf1 and rf2 are the refresh functions used in the power test.

power1 is power the test of run1.

stream11, stream12 and stream13 the are query streams of run1.

refresh1 is the refresh function of run1.

power2 is the power test of run2.

stream21, stream22 and stream23 are the query streams of run2

refresh2 is the refresh function of run1

sum shows the summary for each column.

From Figure 10, we can see: in run1, power test took 12189 seconds; throughput test took 43126 seconds in stream1, 43215 seconds in stream2 and 43971 seconds in stream3; refresh function took 2046 seconds. In run2, power test took 12269 seconds; throughput took 57010 seconds in stream1, 54164 seconds in stream2, and 55921 seconds in stream; refresh function took 2120 seconds.

Performance Metrics for 22 Queries

As in the test1 case, we use the formulas described in section 2.2.7 to get the performance metrics as follows:

S=3, SF=10, Price=$ 67348, Ts=43226 in run1, Ts=36100 in run2.

Run1

TPC-H Power@10GB = 92.25 (query-per-hour)

TPC-H Throughput@10GB = 54.04 (query-per-hour)

QphH@10GB = 70.6 (query-per-hour)

TPC-H Price-per-QphH@10GB = Price ($) / QphH@1GB = 953.94 $

Run2

TPC-H Power@10GB = 90.84 (query-per-hour)

TPC-H Throughput@10GB = 41.68 (query-per-hour)

QphH@10GB = 61.5 (query-per-hour)

TPC-H Price-per-QphH@10GB = Price ($) / QphH@1GB = 1094.56 $

The QphH@10GB in Run2 is less than that in Run1. According to section 2.7, result of Run2 as Figure 11 is considered as the final result of 10GB Oracle test for 22 queries.

QphH@10GB	61.5 query-per-hour
TPC-H Price-per-QphH@10GB	1094.56 $

Figure 11. Metrics of test3 with 22 queries

Performance Metrics for 20 Queries

Since there are problems with query 17 and 20 in DB2, we only calculate 20 queries in following metrics for further comparison.

$$\text{TPC-H Power@Size} = (3600 * SF) / \sqrt[22]{\left(\prod_{i=1}^{i=16} QI(i,0) * \prod_{j=18}^{j=19} QI(i,0) * \prod_{k=21}^{k=22} QI(i,0) * \prod_{l=1}^{l=2} RI(j,0)\right)}$$

TPC-H Througthput@Szie = (S * 20 * 3600) * SF / T_s

$$\text{QphH@size} = \sqrt{\text{Power@size} * \text{Throughput@size}}$$

TPC-H Price-per-QphH@Size = $ / QphH@size

S=3, SF=10, price=$67348 Ts= 42297 in run1, Ts= 54261 in run2.

Run1

TPC-H Power@10GB = 92.54 (query-per-hour)

TPC-H Throughput@10GB = 51.07 (query-per-hour)

QphH@10GB = 68.7 (query-per-hour)

TPC-H Price-per-QphH@10GB = Price ($) / QphH@1GB = 980.32 $

Run2

TPC-H Power@10GB = 91.07 (query-per-hour)

TPC-H Throughput@10GB = 39.8 (query-per-hour)

QphH@10GB = 60.2 (query-per-hour)

TPC-H Price-per-QphH@10GB = Price ($) / QphH@1GB = 1118.74 $

The QphH@10GB in Run2 is less than that in Run1. According to section 2.7, result of Run2 as Figure 12 is considered as the final result of 10GB Oracle test for 20 queries.

QphH@10GB	60.2 query-per-hour
TPC-H Price-per-QphH@10GB	1118.84 $

Figure 12. Metrics of test3 with 20 queries

3.3.2.4 Test4 –10GB Size In DB2

Test4 involves 10GB data in DB2. In this test, we created user table space 16GB, system table space 3GB, temporary table space 16GB, size of total table spaces is 35GB as well. All other database parameters used are the default ones. For instance, page size is 4KB by default. The following Figure 13 shows the execution timing intervals, in seconds, which are used to calculate the Performance Metrics.

From the table shown in Figure 13, we can see that the power test in run1 took 1,750,895 seconds (about 20 days). Recall that power test contains 22 queries and 2 refresh functions. Query 20 took 1,191,487 seconds (about 14 days), query 17 took 533,956 seconds (about 6 days). A query takes definitely longer time in the throughput test than in the power test. For instance, the throughput test took 1.5 times of the power test in test2 (1GB Size in DB2). We estimated the total consuming time of run1 and run2 would be about 100 days. It is not necessary to wait 100 days to get the exact result, since our purpose is to compare two DBMS Oracle and DB2, not the exact query processing time. From the result of the power test in run1, we can see that the performance of DB2 is certainly worse than the performance of Oracle. So, during the throughput test of run1, and the power test and throughput test of run2, we removed query 20 and query 17, and assumed they took the same time with the power test in run1 in the following Figure 13. So query 20 took 1,191,487 seconds, and query 17 took 533,956 seconds in all tests; they are marked with *.

query	run1(seconds)					run2(seconds)				
	power1	throughput1				power2	thoughtput2			
		stream11	stream12	stream13	refresh1		stream21	stream22	stream23	refresh2
q01	1122	2027	2555	2804		1147	2041	3604	2548	
q02	243	1282	1233	1124		182	1362	2238	788	
q03	1004	3809	4023	5051		883	3352	2442	1991	
q04	616	2952	6829	7837		582	3620	4194	9476	
q05	784	6264	5893	2948		745	3156	1524	2990	
q06	591	927	1729	1104		515	718	1017	840	
q07	1371	3694	3723	3888		1174	6500	3552	3707	
q08	963	7103	6922	6344		901	3721	3025	5711	
q09	7844	14291	4923	7879		5369	14649	7964	15414	
q10	767	1757	2320	3047		683	2630	2560	2438	
q11	593	2268	1920	1630		427	972	1281	686	
q12	706	2869	3029	1763		661	1261	2429	1578	
q13	568	1684	1720	1664		406	1867	1071	841	
q14	776	1185	2020	2339		601	1178	3938	2382	
q15	678	2098	2710	2853		547	1200	2831	1459	
q16	112	528	518	317		89	707	238	220	
q17	533956	533956*	533956*	533956*		533956*	533956*	533956*	533956*	
q18	501	3050	3740	3063		535	820	14629	3275	
q19	698	1057	1392	1675		614	726	1208	1513	
q20	1191487	1191487*	1191487*	1191487*		1191487*	1191487*	1191487*	1191487*	
q21	3009	4194	6392	8936		2508	3482	3771	3431	
q22	740	2293	2399	1791		580	1086	1654	2056	
rl	1014					932				
rf2	752					324				
sum	1750895	1790775	1791433	1793500	71715	1745848	1780491	1790613	1788787	68733

Figure 13. Result of test4

Where:

the first column is the query sequence, rf1 and rf2 are the refresh functions used in the power test. power1 is the power test of run1.

stream11, stream12 and stream13 are the query streams of run1.

refresh1 is the refresh function of run1.

power2 is the power test of run2.

stream21, stream22 and stream23 are the query streams of run2

refresh2 is the refresh function of run1

sum shows the summary for each column.

all data with mark * mean inaccurate.

From Figure 10, we can see: in run1, power test took 1750895seconds; throughput test took 1790775seconds in stream1, 1791433 seconds in stream2, and 1793500 seconds in stream3; refresh function took 71715 seconds. In run2, power test took 1745848 seconds; throughput took 1780491 seconds in stream1, 1790613 seconds in stream2, and 1788787 seconds in stream3; refresh function took 68733 seconds.

Performance Metrics for 22 Queries

As in the test1 case, we use the formulas described in section 2.2.7 to get the performance metrics as follows:

S=3, SF=10, Price=$47686, Ts=71715 in run1, Ts=68733 in run2.

Run1

TPC-H Power@10GB = 25.69 (query-per-hour)
TPC-H Throughput @10GB = 1.32* (query-per-hour)
QphH@10GB = 5.8* (query-per-hour)
TPC-H Price-per-QphH@10GB = Price ($) / QphH@1GB = 8193.47* $

Run2

TPC-H Power@10GB = 30.60* (query-per-hour)
TPC-H Throughput@10GB = 1.32* (query-per-hour)

QphH@10GB = 6.3* (query-per-hour)

TPC-H Price-per-QphH@10GB = Price ($) / QphH@1GB = 7509.6* $

Where: the number with * means that it is inaccurate.

The QphH@10GB in Run1 is less than that in Run2. According to section 2.7, the result of Run1 as Figure 14 is considered as the final result of 10GB DB2 test for 22 queries.

QphH@10GB	5.8 query-per-hour
TPC-H Price-per-QphH@10GB	8193.47 $

Figure 14. Metrics of test4 with 22 queries

Performance Metrics for 20 Queries

Since there are problems with query 17 and 20 in DB2, we only calculate 20 queries in following metrics for further comparison.

S=3, SF=10, Price=$47686, Ts= 42297 in run1, Ts= 54261 in run2.

Run1

TPC-H Power@10GB = 45.77 (query-per-hour)

TPC-H Throughput @10GB = 31.74 (query-per-hour)

QphH@10GB = 38.1 (query-per-hour)

TPC-H Price-per-QphH@10GB = Price ($) / QphH@1GB = 1251.27 $

Run2

TPC-H Power@10GB = 55.36 (query-per-hour)

TPC-H Throughput@10GB = 31.43 (query-per-hour)

QphH@10GB = 41.7 (query-per-hour)

TPC-H Price-per-QphH@10GB = Price ($) / QphH@1GB = 1143.27 $

The QphH@10GB in Run1 is less than that in Run2. According to section 2.7, the result of Run1 as Figure 15 is considered as the final result of 10GB DB2 test for 20 queries.

QphH@10GB	38.1 query-per-hour
TPC-H Price-per-QphH@10GB	1251.27 $

Figure 15. Metrics of test4 with 20 queries

4. Comparison

4.1 Comparing Oracle and DB2 in 1GB data

4.1.1 Comparison of Performance Metrics

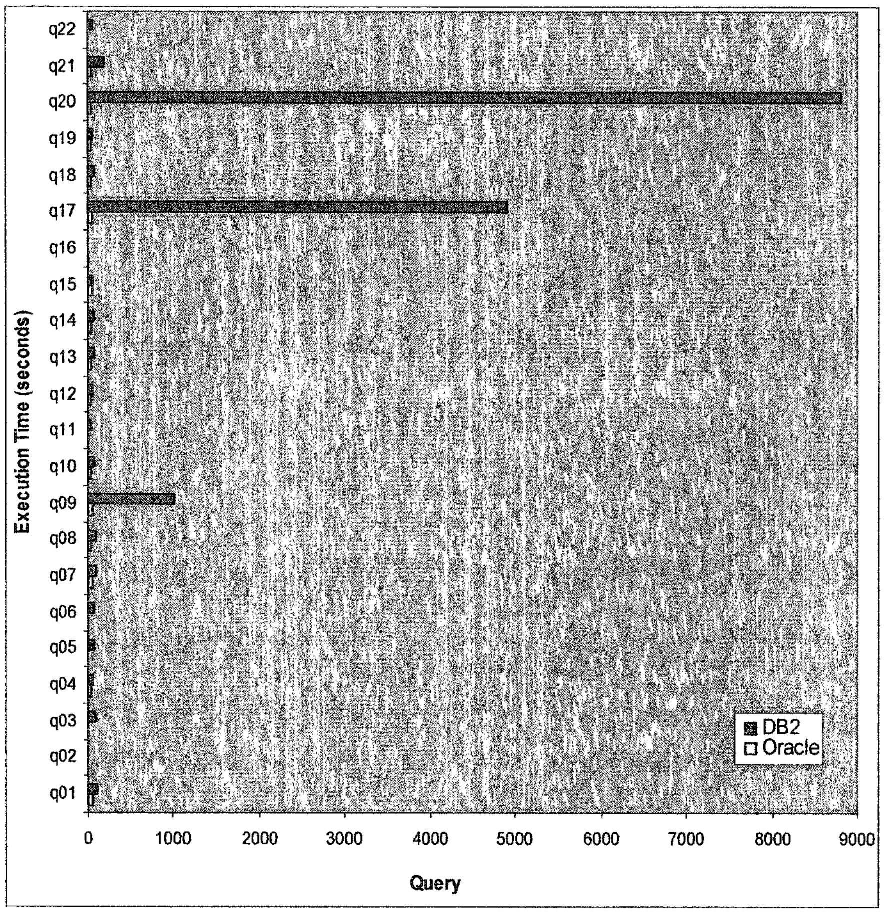
From test1 and test2, we got that the value of QphH@1GB for Oracle is 103.3 query-per-hour, whereas it is 13.3 for DB2. Oracle outperforms DB2 by 7.8 times with 1GB data on Red Hat Linux 7.3. Even when we consider the price factor, where Oracle price is about 1.5 times of DB2's, the value of TPC-H Price-per-QphH@1GB for Oracle is 651.59, whereas it is for DB2 is 3569.3. That is, DB2 costs 5.5 times per QphH@1GB. The comparison of the performance metrics for 1GB data is summarized in Figure 16.

Metrics	**Oracle**	**DB2**
QphH@1GB	103.3	13.3
TPC-H Price-per-QphH@1GB ($)	651.59	3569.3

Figure 16. Comparison of metrics for 1GB data

4.1.2 Comparison of Individual Query with Default Block Size

Figure 17 shows the timing intervals for the power test in run1 with 1GB data for both Oracle and DB2. We can see, the power test of run1 for 1GB data took 708 seconds in Oracle, and 16158 seconds in DB2. Both of refresh functions in Oracle are faster than in DB2. Except for query 2 and query 16, other 20 queries were executed faster in Oracle than in DB2. In particular, query 17 and query 20 are extremely slower in DB2. Query 20 took 33 seconds in Oracle, however it took 8,798 seconds in DB2. Query 17 took 45 seconds in Oracle, however it took 4,905 seconds in DB2. Compared with the largest value (8,798 seconds), the times measured for Oracle are very small, so most of bars in the chart are very small.



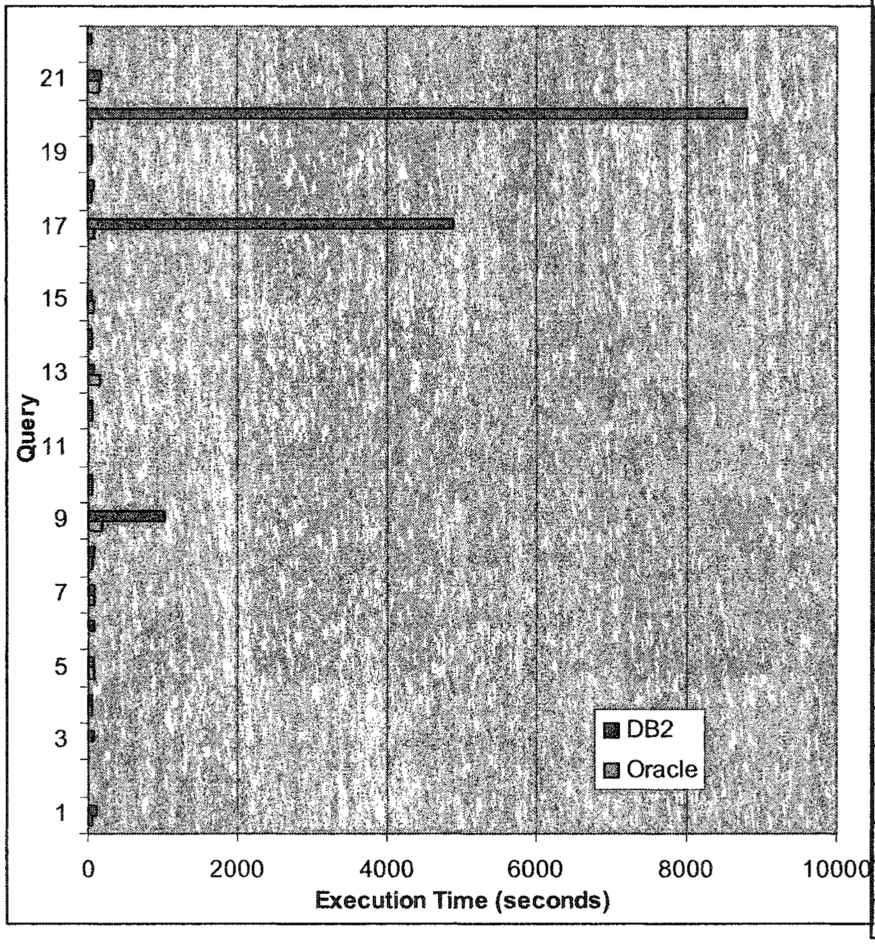
	Oracl	DB2
q01	48	112
q02	16	6
q03	9	90
q04	30	52
q05	8	71
q06	26	74
q07	55	82
q08	34	86
q09	53	1009
q10	32	66
q11	7	38
q12	27	50
q13	28	78
q14	34	65
q15	46	55
q16	11	9
q17	45	4905
q18	34	77
q19	27	61
q20	33	8798
q21	33	187
q22	9	60
rf1	46	90
rf2	17	37
sum	708	16158

Figure 17. Comparison of power1 test for 1GB data

4.1.3 Comparison of Individual Query with 4KB Block Size

As mentioned earlier, we used default setup in our benchmark experiments. For example, the default block size is 8KB in Oracle and 4KB in DB2. Figure 18 shows the timing intervals for the power test in run1 with 1GB data and the block size for both Oracle and

DB2 is 4KB. We can see, the power test of run1 for 1GB data took 1470 seconds in Oracle, and 16158 seconds in DB2. Both of refresh functions in Oracle are faster than in DB2. Except for query 2,4,5,7 and 16, other 17 queries were executed faster in Oracle than in DB2. In particular, query 17 and query 20 are extremely slower in DB2. Query 20 took 47 seconds in Oracle, however it took 8,798 seconds in DB2. Query 17 took 75 seconds in Oracle, however it took 4,905 seconds in DB2. Compared with the largest value (8,798 seconds), the times measured for Oracle are very small, so most of bars in the chart are very small.



	Oracl	DB2
q01	53	112
q02	11	6
q03	13	90
q04	55	52
q05	91	71
q06	37	74
q07	84	82
q08	51	86
q09	183	1009
q10	53	66
q11	11	38
q12	48	50
q13	162	78
q14	46	65
q15	82	55
q16	15	9
q17	75	4905
q18	49	77
q19	45	61
q20	47	8798
q21	154	187
q22	13	60
rf1	49	90
rf2	28	37
sum	1470	16158

Figure 18. Comparison of 4KB block size for 1GB data

4.2 Comparing Oracle and DB2 in 10GB data

4.2.1 Comparing Performance Metrics

From test3 and test4, we got the summary results as Figure 19. With 22 queries, the value of QphH@10GB for Oracle is 61.5 query-per-hour, whereas it is 5.8 in DB2. As the section 3.3.2.4 described, we assumed that the query 20 and query 17 in all DB2 tests took the same time with the power test in run1, actually they certainly would take more time in the throughput test than in the power test. As a result, Oracle outperforms DB2 by 10.6 times on Red Hat Linux 7.3. Considering again the price factor, the value of TPC-H Price-per-QphH@10GB in Oracle is 1094.56, whereas it is more than 8193.47 in DB2. In this case, DB2 costs at least 7.5 times per QphH@10GB.

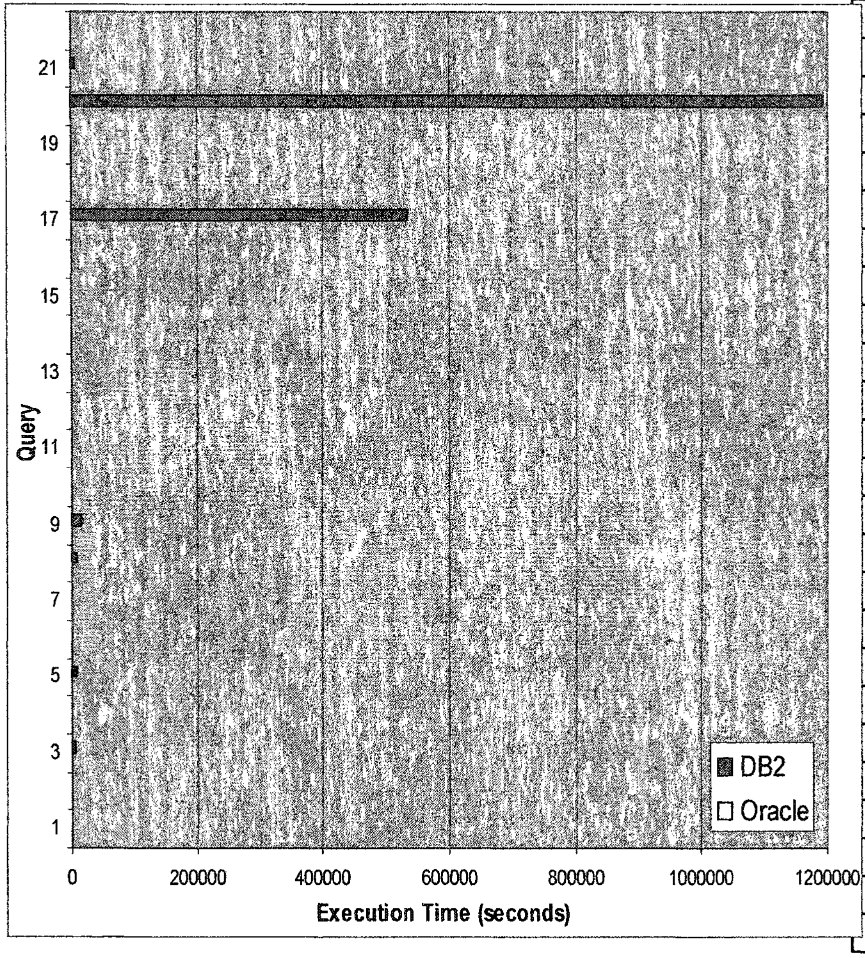
If we removed those two queries, which caused problems in DB2 experiment, then we calculated the metrics as Figure 19 shown. The value of QphH@10GB for Oracle is 60.2 query-per-hour, whereas it is 38.1 in DB2. Oracle still outperforms DB2 by 1.6 times on Red Hat Linux 7.3. Considering again the price factor, the value of TPC-H Price-per-QphH@10GB in Oracle is 1118.84, whereas it is more than 1251.27 in DB2. DB2 still costs 12% more per QphH@10GB.

# of Queries & DBMS / **Metrics**	**22 Queries**		**20 Queries**	
	Oracle	**DB2**	**Oracle**	**DB2**
QphH@10GB	61.5	5.8	60.2	38.1
TPC-H Price-per-QphH@10GB ($)	1094.56	8193.47	1118.84	1251.27

Figure 19. Comparison of metrics for 10GB data

4.2.2 Comparing Individual Query

Figure 20 shows the timing intervals of the power test in run1 for 10GB data in Oracle and DB2. We can see the power test of run1 for 10GB data took 12,189 seconds (about 3.4 hours) in Oracle, 1,750,895 seconds (about 20 days) in DB2. Both of refresh functions in Oracle are faster than in DB2, refresh function 1 took 210 seconds in Oracle and 1014 seconds in DB2; refresh function 2 took 169 seconds in Oracle and 752 seconds in DB2. All 22 queries in Oracle were executed faster than in DB2. Especially query 17 and query 20 were extremely slower in DB2. Query 20 took 1,191,487 seconds (about 13.8 days) in DB2, however it took 313 seconds (5.2 minutes) in Oracle. Query 17 took 533,956 seconds (about 6 days) in DB2, however it took 519 seconds (8.6 minutes) in Oracle. Compared with the largest value (1,191,487 seconds), the times measured for Oracle are very small, so we can see only a few large values for DB2 in this chart.



	Oracle	DB2
q01	472	1122
q02	134	243
q03	688	1004
q04	380	616
q05	680	784
q06	248	591
q07	672	1371
q08	370	963
q09	1594	7844
q10	521	767
q11	75	593
q12	496	706
q13	1012	568
q14	297	776
q15	542	678
q16	169	112
q17	519	533956
q18	338	501
q19	784	698
q20	313	1191487
q21	1381	3009
q22	125	740
rf1	210	1014
rf2	169	752
sum	12189	1750895

Figure 20. Comparison of power1 test for10GB data

4.3 Comparing Oracle on Linux and Windows

Our TPC-H benchmark experiment was done on Red Hat Linux 7.3 operating system. A similar study on Windows 2000 platform was done in parallel [4].

4.3.1 Comparing Performance Metrics in Oracle

From our Oracle experiment results on Linux presented in section 3 and the results on Windows 2000 [4], we summarize the performance metrics for Oracle 1GB and 10GB data on Linux and Windows as Figure 21.

The value of QphH@1GB on Linux is 103.3 (query-per-hour), whereas it is 28.4 on Windows; the value of QphH@10GB on Linux is 61.5, whereas it is 19.7 on Windows. The performance of Oracle for both 1GB and 10GB data on Linux is better than on Windows. For 1GB data, Oracle outperforms 3.6 times on Linux than Windows. For 10GB data, Oracle outperforms 3.1 times on Linux than Windows.

Metrics	**Linux**	**Windows**
QphH@1GB	103.3	28.4
TPC-H Price-per-QphH@1GB ($)	651	2371
QphH@10GB	61.5	19.7
TPC-H Price-per-QphH@10GB ($)	1094	3419

Figure 21. Comparison of Oracle performance metrics on Linux and Windows

4.3.2 Comparing Individual Test in Oracle

4.3.2.1 Comparing Individual Test in Oracle for 1GB data

Figure 22 illustrates a summary result of individual Oracle test for 1GB data on Linux and Windows. For example, in run1, the power test took 708 seconds on Linux, whereas 1499 seconds on Windows; the stream1 took 1,784 seconds on Linux, whereas 16,279 seconds on Windows; the stream2 took 1,679 seconds on Linux, whereas 15,277 seconds on Windows; refresh function took 795 seconds on Linux, whereas 80 seconds on

Windows. We can see all queries streams on Linux are faster than on Windows, but refresh functions on Windows are faster than on Linux.

Oracle (1GB)	run1 (seconds)				run2 (seconds)			
	power1	throughput1			power2	thoughtput2		
		stream11	stream12	refresh1		stream21	stream22	refresh2
Linux	708	1784	1679	795	833	1873	1948	1419
Windows	1499	16279	15277	80	1486	15123	15275	74

Figure 22. Comparison of Oracle performance for 1GB data on Linux and Windows

4.3.2.2 Comparing Individual Test in Oracle for 10GB data

Figure 23 illustrates a summary result of individual Oracle test for 10GB data on Linux and Windows. For example, in run1, the power test took 12,189 seconds on Linux, whereas 21,823 seconds on Windows; the stream1 took 43,126 seconds on Linux, whereas 304,918 seconds on Windows; the stream2 took 43,215 seconds on Linux, whereas 334,313 seconds on Windows; the stream3 took 43,971 seconds on Linux, whereas 309,864 seconds on Windows; the refresh function took 2,046 seconds on Linux, whereas 1,534 seconds on Windows. We can see all queries streams on Linux are faster than on Windows, but refresh functions on Windows are faster than on Linux.

Oracle (10GB)	run1(seconds)					run2(seconds)				
	power1	throughput1				power2	thoughtput2			
		stream11	stream12	stream13	refresh1		stream21	stream22	stream23	refresh2
Linux	12189	43126	43215	43971	2046	12269	57010	54164	55921	2120
Windows	21823	304918	334313	309864	1534	21494	333217	339696	334083	1570

Figure 23. Comparison of Oracle performance for 10GB data on Linux and Windows

4.4 Comparing DB2 on Linux and Windows

4.4.1 Comparing Performance Metrics in DB2

Figure 24 illustrates the performance metrics for DB2 1GB and 10GB data on Linux and Windows. The value of QphH@1GB on Linux is 13.3 (query-per-hour), whereas it is 10.4 on Windows. DB2 for 1GB data on Linux outperforms on Windows by 28%.

With 22 queries, value of QphH@10GB on Linux is 5.8, whereas it is 4.7 on Windows. DB2 for 10GB data on Linux outperforms on windows by 23%. As mentioned earlier, we experienced difficulty to execute query 17 and 20 for 10G data on both Windows and Linux. We use 533,956 seconds (148 hours) for Query 17 and 1,191,487 seconds (330 hour) for query 20 in all queries streams on Linux and Windows to calculate these metrics.

If we remove queries 17 and 20, which caused problems in both Linux and Windows environment, to calculate the metrics, then the value of QphH@10GB is 38.1 query-per-hour on Linux, whereas it is 18.7 on Windows. DB2 outperforms 2 times on Linux.

# of Queries & OS / **Metrics**	**22 Queries**		**20 Queries**	
	Windows	**Linux**	**Windows**	**Linux**
QphH@1GB	10.4	13.3		
TPC-H Price-per-QphH@1GB ($)	4585	3569		
QphH@10GB	4.7	5.8	18.7	38.1
TPC-H Price-per-QphH@10GB ($)	10146	8193	2542	1251

Figure 24. Comparison of DB2 performance metrics on Linux and Windows

4.4.2 Comparing Individual Test in DB2

4.4.2.1 Comparing Individual Test in DB2 for 1GB data

Figure 25 illustrates a summary result of individual DB2 test for 1GB data on Linux and Windows. For example, in run1, power test took 16158 seconds on Linux, whereas 20557 seconds on Windows; stream1 took 32318 seconds on Linux, whereas 41096 seconds on Windows; stream2 took 33709 seconds on Linux, whereas 38587 seconds on Windows; the refresh function took 32906 seconds on Linux, whereas 1185 seconds on Windows. We can see all queries streams in DB2 for 1GB data on Linux are faster than on Windows, but refresh functions on Windows are faster than on Linux.

DB2 (1GB)	run1 (seconds)				run2 (seconds)			
	power1	throughput1			power2	thoughtput2		
		stream11	stream12	refresh1		stream21	stream22	refresh2
Linux	16158	32318	33709	32906	13725	29367	29750	30103
Windows	20557	41096	38587	1185	18129	41477	38338	1044

Figure 25. Comparison of DB2 performance for 1GB data on Linux and Windows

4.3.2.1 Comparing Individual Test in DB2 for 10GB data

Figure 26 illustrates a summary result of individual DB2 test for 10GB data on Linux and Windows. As mentioned earlier, we experienced difficulty to execute query 17 and query 20 in DB2 for 10GB data on both Linux and Windows. In this comparison, we removed those two queries, so only 20 queries and 2 refresh functions are calculated in Figure 26. For example, in run1, the power test contains 20 queries and 2 refresh functions took 25,452 seconds on Linux, whereas 43,095 seconds on Windows; stream1 took 65,332 seconds on Linux, whereas 191112 seconds on Windows; stream2 took 65,990 seconds on Linux, whereas 174,844 seconds on Windows; stream3 took 68,057 seconds on Linux, whereas 189,704 seconds on Windows; refresh function took 71,715seconds on Linux, whereas 193,079 seconds on Windows. We can see that every individual test in DB2 for 10GB data on Linux is faster than on Windows.

DB2	run1(seconds)					run2(seconds)				
(10GB)	power1	throughput1				power2	thoughtput2			
		stream11	stream12	stream13	refresh1		stream21	stream22	stream23	refresh2
Linux	25452	65332	65990	68057	71715	20405	55048	65170	63344	68733
Windows	43095	191112	174844	189704	193079	50071	184303	189690	177230	190536

Figure 26. Comparison of DB2 performance for 10GB data on Linux and Windows

5. Performance Tuning

There are many factors that affect database performance, for instance, hardware, operating system, and the different setup parameters of operating system and the database management system. It is not difficult to understand how the performance would be affected by hardware, for example, a computer with more CPUs and more memories is definitely faster. Using the same hardware, performance is different with the different Operating Systems. From section 4, we already know that the performance of both Oracle and DB2 is better on Linux than on Windows 2000 for. Here, we just describe how the database performance is affected by the different setup parameters of operating systems, such as swap space and shared memory, and by different setup parameters of database system, such as, data block size, statistics collection, size of undo table space, and size of temporary table space.

5.1 Oracle Database Performance Tuning

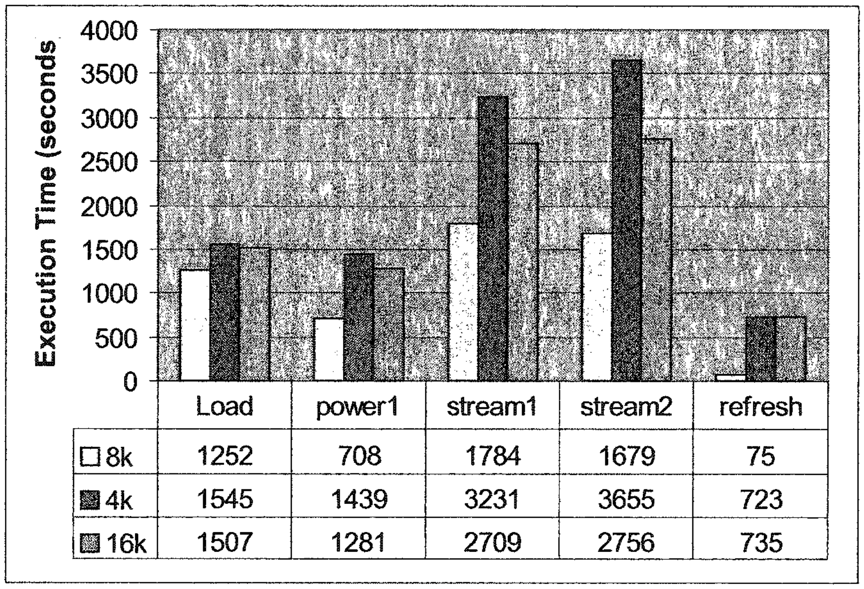
5.1.1 Data Block Size

The DB_BLOCK_SIZE initialization parameter specifies the standard block size for the database. This block size is used for the *system* table space and by default in other table spaces [9]. The database block size must be a multiple of the operating system's block size [7]. In Red Hat Linux 7.3, the operating system's block size is 4K.

A larger data block size provides greater efficiency in disk and memory I/O (access and storage of data). How large a database block size should be is application dependent.

Usually a lager data block size is good for a larger OLAP database, and a smaller data block size is good for a small OLTP database. Also the block size cannot be changed after the database is created, except by re-creating the database.

We test three different block sizes, 4KB, 8KB and 16KB with 1GB data size. Figure 27 shows the timing intervals, in seconds. The timing intervals for load test contain loading data and updating statistics. Power1 is the power test in run1. Stream1, stream2 and refresh are the query streams and refresh functions in run1. From Figure 27, we can see how data block size affects the performance. 8KB data block size provides the best performance in our 1GB database for load test, power test and throughput test. 16KB data block size is better than 4KB data block size. Especially refresh functions in throughput test, a pair of refresh functions took 75 seconds with 8KB data block size, 723 seconds with 4KB data block size, and 735 seconds with 16KB data block size.



	Load	power1	stream1	stream2	refresh
□ 8k	1252	708	1784	1679	75
▩ 4k	1545	1439	3231	3655	723
▨ 16k	1507	1281	2709	2756	735

Figure 27. Different block sizes in Oracle

5.1.2 Statistics Collection

To execute a SQL statement, Oracle may have to perform many steps. Each of these steps either retrieves rows of data physically from the database or prepares them in some way for the user issuing the statement. The combination of the steps Oracle uses to execute a statement is called an execution plan [7]. Many different execution plans to execute a SQL statement often exist, for example, by varying the order in which tables or indexes are accessed [9]. However, how does Oracle determine which execution plan should be used for a given SQL? The cost-based optimizer (CBO) uses statistics to estimate the number of disk I/O, CPU time, and memory are required to execute the query using a particular execution plan and determines which execution plan is most efficient by considering available access paths and the statistics for the schema objects (tables or indexes) accessed by the SQL query [7]. The statistics are stored in the data dictionary and can be generated with the DBMS_STATS package.

We have done two power tests to see how much statistics collection improves performance in our 1GB database. After loading the data, if the statistics information was collected, the power test in run1 took 708 seconds; if no statistics was collected, the power test in run1 took 1,779 seconds as shown in Figure 28. The power test with statistics collection outperforms by 2.5 times. So, with the statistics collection, Oracle can choose a better execution plans.

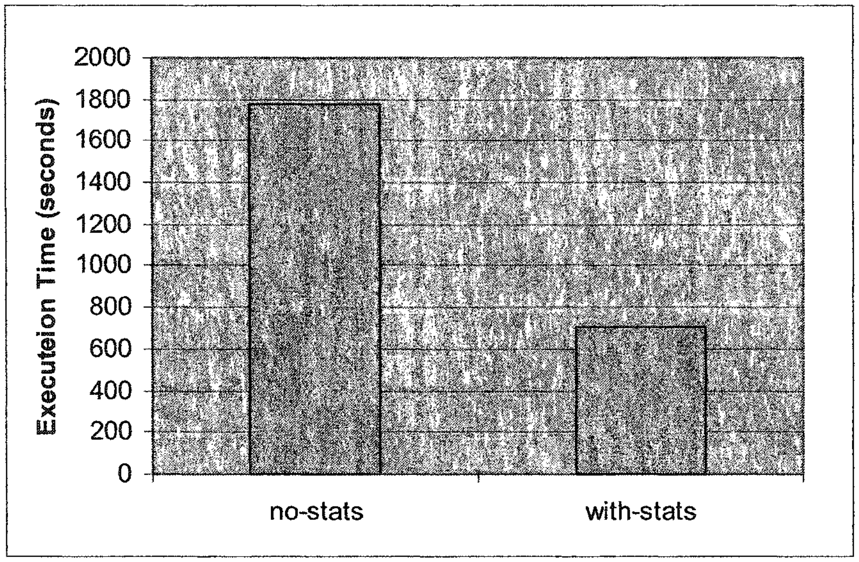
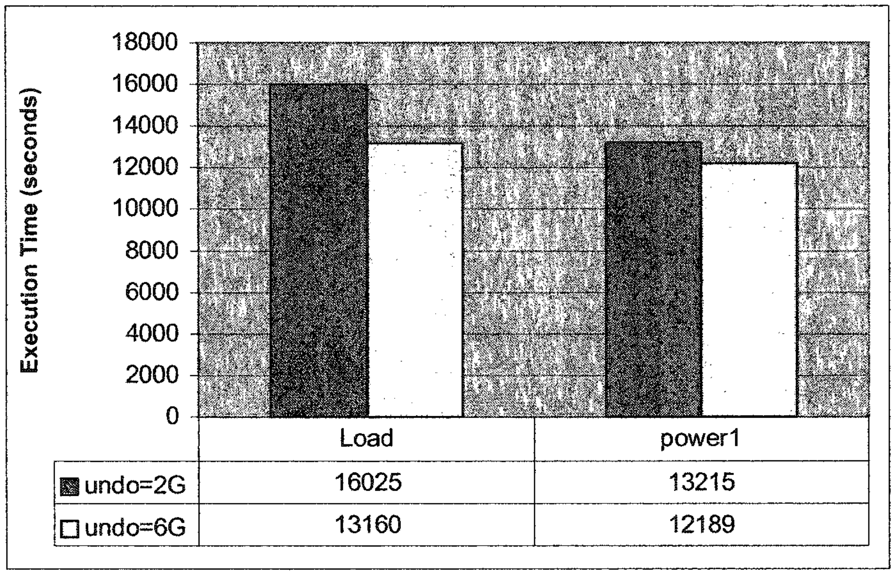


Figure 28. Statistics collection in Oracle

5.1.3 Undo Table Space

Oracle9i provides automatic undo management to replace rollback segment. Database Administrators (DBA) often have a hard time to manually tune the rollback segments. Automatic undo management completely automates the management of undo data and significantly simplifies database management and removes the need for any manual tuning of undo (rollback) segments [7]. A database running in automatic undo management mode transparently creates and manages undo segments [8]. However, how large undo table space should be? It really depends on the specific database application. For example, how many transactions are running at the same time? How much data are accessed at the same time? If undo table space is too small, we may get error message "snapshot is too old" since the records to keep read consistence in unto table space are overwritten. With the smaller undo table space, even through we may not get any error message, it may have an impact on the performance.

Figure 29 illustrates the performance with 2GB and 6GB undo table space in our 10GB test database. We can see the performance with 6GB undo table space is better than that with 2GB table space. However, undo table space does not affect the performance as much as the block size and statistics do.



	Load	power1
▨ undo=2G	16025	13215
□ undo=6G	13160	12189

Figure 29. Undo table space in Oracle

5.1.4 Temporary Table Space

When processing queries, Oracle often requires temporary workspace for intermediate stages of queries parsing and execution. Typically, Oracle requires a temporary segment as a work area for sorting [7]. Oracle does not create a segment if the sorting operation can be done in memory or if Oracle finds some other way to perform the operation using indexes. Also, when collecting statistics, Oracle uses temporary table space for intermediate stages. The size of temporary table space may not affect the performance, however, we may get error message with a small temporary table space.

At the beginning, we created 500MB temporary table space for 1GB database. When loading data and starting collecting statistics, we got the error message: “ORA-01652: unable to extend temp segment by 128 in table space TEMP”. After we changed the temporary table space to 1000MB, it worked well.

5.1.5 Swap Space and Shared Memory

With 10GB data, power test was completed smoothly, but we experienced problems in the throughput test when we have 4 sessions running at the same time. The error message was "ORA-00601: cleanup lock conflict" and the database server was down. Unlike what the above error message indicates, there should not be any locks since the session that refresh data was already done successfully, we just had "select" queries and no transactions changing the database state. We repeated this test 3 times, and it always failed with the same error message. When we ran those three streams which contained 22 queries separately, there was no error. When we removed 2 queries from 4 streams, it worked fine as well. However, problem happened only when we concurrently ran 4 streams containing 22 queries.

We tried to find help from Oracle website or Oracle forums. There are some similar problems reported, but not for Oracle 9.0.1 on Linux. There are lots of patches for Oracle 9.0.1 on Linux, but none for this particular problem.

Finally we found that the Linux system we used had too small swap space and shared memory. The swap space was 514,072 bytes. We increased the temporary swap space 899,992 bytes. The shared memory was 33,554,432 bytes, which we temporarily increased the Linux system kernel parameter *shmmax* to 1,073,741,824. After these changes, the Linux system worked fine. Here are the commands that we used to do these:

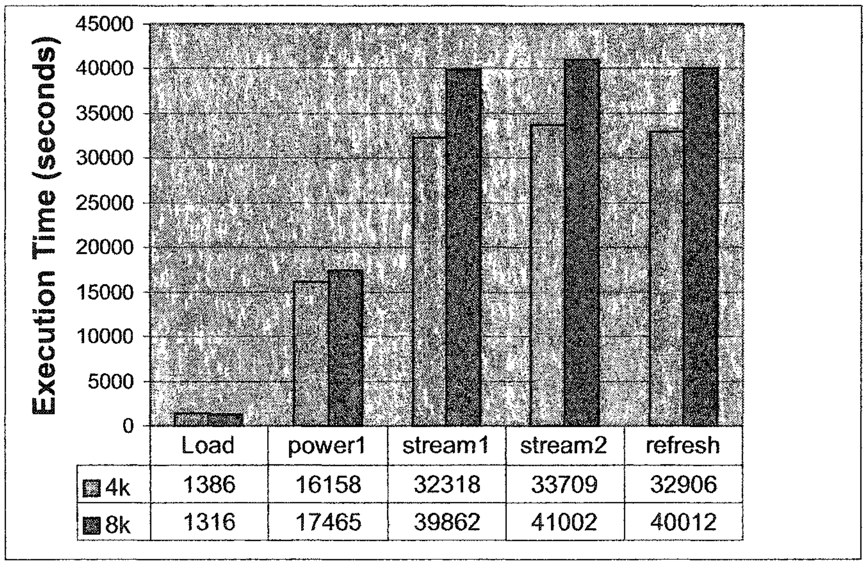
```
As root:
#  cat /proc/swaps          //Check swap space
Filename                Type        Size    Used   Priority
/dev/hda2               partition   514072  0      -1
#dd if=/dev/zero of=tmpswap bs=1k count=900000
#chmod 600 tmpswap
#mkswap tmpswap
#swapon tmpswap
#cat /proc/swaps
Filename                Type        Size    Used   Priority
/dev/hda2               partition   514072  0      -1
/root/tmpswap             file      899992  0      -2
# cat /proc/sys/kernel/shmmax
33554432
# echo `expr 1024 \* 1024 \* 1024` > /proc/sys/kernel/shmmax
# cat /proc/sys/kernel/shmmax
1073741824
```

5.2 DB2 Database Performance Tuning

5.2.1 Page Size

When we created the test database in DB2, there were three table spaces by default: one catalog table space, one user table space, and one system temporary table space. The page size for every table space was 4KB, by default. We could not change the page size for any table space after they had been created except for recreating database. Also we could not drop catalog table space and system temporary table space. However we could drop user table space and re-created it with different page size.

Figure 30 illustrates the performance with 4KB and 8KB page sizes. We can see that the performance of 4KB page size is better than the 8KB page size for both power test and throughput test. The database load time is almost the same. But, as mentioned earlier, database load timing is inaccurate since timing for checking integrity constraints and gathering statistics was recorded manually, as it was not supported by the system. In the 4KB page size test, the page sizes for all table spaces were 4KB. However, in the 8KB page size experiment, the page size for the user table space was 8KB, the page sizes for the catalog table space and system temporary table space was 4KB. In addition, we were not able to test 2KB page size because the minimum page size is 4KB.



	Load	power1	stream1	stream2	refresh
▨ 4k	1386	16158	32318	33709	32906
■ 8k	1316	17465	39862	41002	40012

Figure 30. Different page sizes in DB2

5.2.2 Statistics Collection

Statistics is very important for query optimization in database systems to improve system performance. Many different execution plans to execute a SQL statement often exist, for example, by varying the order in which tables or indexes are accessed. However, how does DB2 determine which execution plan should be used for a specific SQL? DB2 optimizer uses statistics to estimate the number of disk I/O, CPU time, and memory required to execute a SQL statement using a particular execution plan and determines which execution plan is most efficient by considering available access paths and by factoring in information based on statistics for the schema objects (tables or indexes) accessed by the SQL statement. If the statistics information is not up-to-date, DB2 may choose a worse execution plan.

Figure 31 illustrates the comparison of the power test with 1GB data. We have done two power tests to see how much statistics collection improves performance in our 1GB database. After loading the data, if the statistics information was collected, power test in run1 took 16158 seconds; if no statistics was collected, power test in run1 took 19389 seconds as shown in Figure 31. The power test with statistics collection outperforms by 20%. So, with the statistics collection, DB2 can choose a better execution plans. Compared to Oracle, in which power test with statistics collection outperforms by 2.5 times, the statistics information affects less.

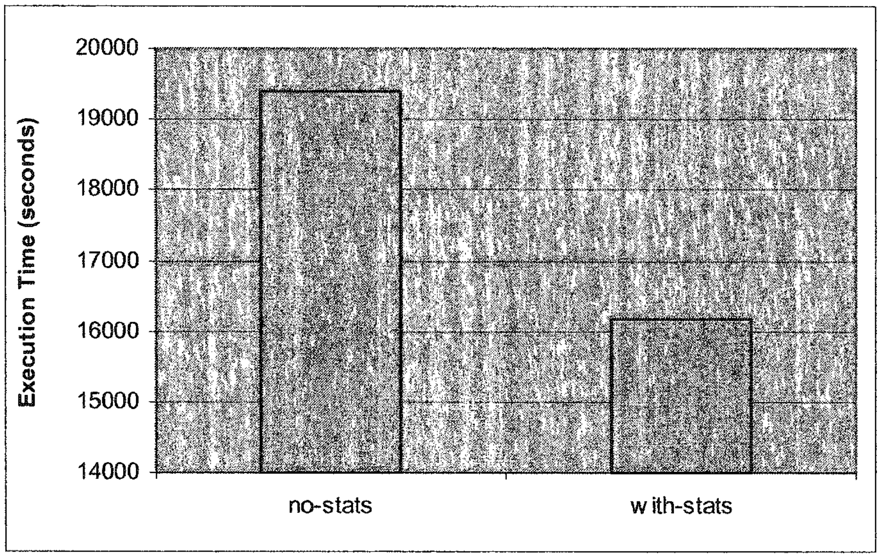


Figure 31. Statistics collection in DB2

5.2.3 Temporary Table Space

A system temporary table space is used to store system temporary tables. When a database is created, one of the three of default table spaces defined is a system temporary table space. A user temporary table space is used to store declared temporary tables. Declared temporary tables are stored in system temporary tables if there is no user temporary table spaces created. By default, no user temporary table spaces are created.

At the beginning, we created 500MB system temporary table space for 1GB database. There was no problem for loading data, but we got error message "SQL0289N Unable to allocate new pages in table TEMP space SQLSTATE=57011" during the power test. We then changed the system temporary table space to 1000MB; then, it worked well. Like Oracle, the size of temporary table space may not affect the performance, however, we may get error message with a small temporary table space.

5.2.4 Swap Space and Shared Memory

With 10GB database test, there are problems with queries 17 and 20. In the power test, query 17 took about 6 days, and query 20 took about 14 days. It is definitely not normal. We tried to increase the swap space and the shared memory, as we did in our Oracle experiment, but this did not help in our DB2 experiment.

6. Conclusions

In our TPC-H benchmark experiment, Oracle demonstrated better performance than DB2 on our desktop computer on Red Hat Linux 7.3, irrespective of whether it has a 1GB or 10GB data size.

- With 1GB data, Oracle outperforms DB2 by 7.8 times, and DB2 costs 5.5 times per QphH@1GB. Every individual query and refresh function except for query 2 and 16 is faster in Oracle than in DB2.
- With 10GB data and 22 queries, Oracle outperforms DB2 by 10.6 times, and DB2 costs 7.5 times per QphH@10GB. Every individual query and refresh function is faster in Oracle than in DB2.
- With 10GB data and 20 queries, Oracle outperforms DB2 by 1.6 times, and DB2 costs 12% more per QphH@10GB.

Both Oracle and DB2 show better performance on Red Hat Linux 7.3 than on Windows2000 operating system.

- With 1GB data, Oracle outperforms 3.6 times on Linux than Windows.
- With 10GB data, Oracle outperforms 3.1 times on Linux than Windows.
- In Oracle, all queries streams on Linux are faster than on Windows, but refresh functions on Windows are faster than on Linux for both 1GB and 10GB data.
- DB2 for 1GB data on Linux outperforms on Windows by 28%.
- With 22 queries, DB2 for 10GB data on Linux outperforms on windows by 23%.
- With 20 queries, DB2 for 10GB data on Linux outperforms on windows by 2 times.

- In DB2, all queries streams on Linux are faster than on Windows, but refresh functions on Windows are faster than on Linux for 1GB database.
- In DB2, all queries streams and refresh functions on Linux are faster than on Windows for 10GB database.

On database performance tuning, we can get the following conclusions:

- Both Oracle and DB2, the default data block or page size demonstrates the best performance in 1GB database. In Oracle, the default data block size is 8KB. In DB2, the default page size is 4KB.
- Statistics information can improve the performance of Oracle and DB2. In Oracle 1GB database, the power test with statistics collection outperforms by 2.5 times. In DB2 1GB database, the power test with statistics collection outperforms by 20%.
- The size of temporary table space may not affect the performance, however, we may get error message with a small temporary table space.

The user interface of Oracle is friendlier than that of DB2 on both Linux and Windows. The installation of DB2 was easier and faster than that of Oracle on Linux. The installation of Oracle and DB2 software on Windows is easier and faster than on Linux.

The price of Oracle 9i is about $63,348, and the price of DB2 7.2 is about $43,686. The price of Windows 2000 is about $200, and Red Hat 7.3 is free.

What we can learn from this Project

From this project, we learned a few important lessons.

- First, we learned how to install and manipulate the commercial DBMS: DB2 Universal Database 7.2 and Oracle 9i and utilities provided by them. For example, we can use Oracle *Database Configuration Assistant* to create a database, whereas we can use DB2 *control center* to create a new user table space with a different page size.
- Second, we learned how to use Linux *crontab* to schedule the tasks to run automatically in the background of the system. Also, we learned how to add a second hard drive in Linux operating system [5].
- Third, we learned how to tune DBMS performance. For example, the project can help us realize how data block sizes, statistics information, undo table spaces and temporary table spaces affect DBMS performance.
- Finally, this project can help us realize which DBMS has better performance over which operating systems. The decision about choosing appropriate DBMS on certain operating systems can be made. Oracle9i is better than DB2 7.3. Linux7.3 is better than Windows 2000.

References

[1] IBM, *IBM announces the world's first 10 terabyte TPC-H benchmark result* December 2002, http://www-3.ibm.com/software/data/highlights/db2leads-tpch.html

[2] Oracle Corporation, *Oracle Announces New World Record 3-Terabyte TPC-H Benchmark; Demonstrates Oracle9i Database Leadership In Running Very Large Databases*, August 2002, http://biz.yahoo.com/iw/020827/045880.html .

[3] TPC Organization, *TPC BenchMark^{TM}H (decision support) Standard Specification Reversion 1.5.0*, July 2002.

[4] Jing Zhou, Concordia University, Master major repot, *Database Performance Analysis and Tuning: A Comparative Study of TPC-H Benchmark on Oracle and DB2*, March 2003.

[5] Werner Puschitz, *9i Installation on Red Hat Linux 7.1, 7.2, .3 8.0, and on Red Hat Advanced Server 2.1* http://www.puschitz.com/OracleOnLinux.shtml#12 .

[6] Kevin Czap and Ian Shields, *How to install and configure DB2 for Linux and the Java Runtime Environment.* http://www-106.ibm.com/developerworks/library/l-ss-db2/ .

[7] Oracle Corporation, Oracle9i *Database Concepts Release 2 (9.2)* March 2002.

[8] Oracle Corporation, Oracle9i *Database Administrator's Guide Release 2 (9.2),* March 2002.

[9] Oracle Corporation, *Database Performance Tuning Guide and Reference Release 2 (9.2)*, October 2002.

[10] Mendel Leo Cooper, *Adding a Second IDE Hard Drive to Your System* http://www.linuxgazette.com/issue38/cooper.html .

Appendices

1. Pricing Summary Report Query (Q1)

This query reports the amount of business that was billed, shipped, and returned.

Business Question

The Pricing Summary Report Query provides a summary pricing report for all lineitems shipped as of a given date. The date is within 60-120 days of the greatest ship date contained in the database. The query lists totals for extended price, discounted extended price, discounted extended price plus tax, average quantity, average extended price, and average discount. These aggregates are grouped be RETURNFALG and LINESTATUS, and listed in ascending order of RETURNFLAG and LINESTATUS. A count of the number of linetems in each group is included.

Functional Query Definition

```
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= to_date(date '1998-12-01' - interval '104' day (3))
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
```

2. Minimum Cost Supplier Query (Q2)

This query finds which supplier should be selected to place an order for a given part in a given region.

Business Question

The Minimum Cost Supplier Query finds, in a given region, for each part of a certain type and size, the supplier who can supply it at minimum cost. If server suppliers in that region offer the desired part type and size at the same (minimum) cost, the query lists the parts from suppliers with the 100 highest account balances. For each supplier, the query lists the supplier's account balance, name and nation; the part's number and manufacturer; the supplier's address, phone number and comment information.

Functional Query Definition

```
select
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
from
    part,
    supplier,
    partsupp,
    nation,
    region
where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = 6
    and p_type like '%TIN'
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'MIDDLE EAST'
    and ps_supplycost = (
        select
            min(ps_supplycost)
        from
            partsupp,
            supplier,
            nation,
            region
        where
            p_partkey = ps_partkey
            and s_suppkey = ps_suppkey
            and s_nationkey = n_nationkey
            and n_regionkey = r_regionkey
            and r_name = 'MIDDLE EAST'
    )
    and rownum < 101
order by
    s_acctbal desc,  n_name,  s_name,  p_partkey;
```

3. Shipping Priority Query (Q3)

This query retrieves the 10 unshipped orders with the highest value.

Business Question

Shipping Priority Query retrieves the shipping priority and potential revenue, defined as the sum of l_extendedprice * (1-l_discount), of the orders having the largest revenue among those that had not been shipped as of a given date. Orders are listed in decreasing order of revenue. If more than 10 unshipped orders exist, only the 10 orders with the largest revenue are listed.

Functional Query Definition

```
select
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'MACHINERY'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < to_date (date '1995-03-26')
    and l_shipdate > to_date (date '1995-03-26')
    and rownum < 11
group by
    l_orderkey,
    o_orderdate,
    o_shippriority
order by
    revenue desc,
    o_orderdate;
```

4. Order Priority Checking Query (Q4)

This query determines how well the order priority system is working and gives an assessment of customer satisfaction.

Business Question

The Order Priority Checking Query counts the number of orders ordered in a given quarter of a given year in which at least one lineitem was received by the customer later than its committed date. The query lists the count of such orders for each order priority sorted in ascending priority order.

Functional Query Definition

```
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= to_date (date '1997-01-01')
    and o_orderdate < to_date (date '1997-01-01' + interval '3' month)
    and exists (
        select
            *
        from
            lineitem
        where
            l_orderkey = o_orderkey
            and l_commitdate < l_receiptdate
    )
group by
    o_orderpriority
order by
    o_orderpriority;
```

5. Local Supplier Volume Query (Q5)

This query lists the revenue volume done through local suppliers.

Business Question

Local Supplier Volume Query lists for each nation in a region the revenue volume that resulted from lineitem transaction in which the customer ordering parts and the supplier filling them were both within that nation. The query is run in order to determine whether to institute local distribution centers in a given region. The query considers only parts ordered in a given year. The query displays the nations and revenue volume in descending order by revenue. Revenue volume for all qualifying lineitems in a particular nation is defined as sum(l_extendedprice * (1 - l_discount)).

Functional Query Definition

```
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'MIDDLE EAST'
    and o_orderdate >= to_date (date '1993-01-01')
    and o_orderdate < to_date (date '1993-01-01' + interval '1' year)
group by
    n_name
order by
    revenue desc;
```

6. Forecasting Revenue Change Query (Q6)

This query quantifies the amount of revenue increase that would have resulted from eliminating certain company wide discounts in a given percentage range in a given year. Asking this type of "what if" query can be used to look for ways to increase revenues.

Business Question

Forecasting Revenue Change Query considers all the lineitems shipped in a given year with discounts between DISCOUNT-0.01 and DISCOUNT+0.01. The query lists the amount by which the total revenue would have increased if these increase is equal to the sum of (l_extendedprice * l_discount) for all lineitems with discounts and quantities in the qualifying range.

Functional Query Definition

```
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= to_date (date '1993-01-01')
    and l_shipdate < to_date (date '1993-01-01' + interval '1' year)
    and l_discount between 0.06 - 0.01 and 0.06 + 0.01
    and l_quantity < 24;
```

7. Volume Shipping Query (Q7)

This query determines the value of goods shipped between certain nations to help in the re-negotiation of shipping contracts.

Business Question

Volume Shipping Query finds, for two given nations, the gross discounted revenues derived from lineitems in which parts were shipped from a supplier in either nation to a customer in the other nation during 1995 and 1996. The query lists the supplier nation, the customer nation, the year, and the revenue from shipments that took place in that year. The query orders the answer by Supplier nation, Customer nation, and year (all ascending).

Functional Query Definition

```
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from
    (
        select
            n1.n_name as supp_nation,
            n2.n_name as cust_nation,
            extract(year from l_shipdate) as l_year,
            l_extendedprice * (1 - l_discount) as volume
        from
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2
        where
            s_suppkey = l_suppkey
            and o_orderkey = l_orderkey
            and c_custkey = o_custkey
            and s_nationkey = n1.n_nationkey
            and c_nationkey = n2.n_nationkey
            and (
                ( n1.n_name = 'KENYA' and n2.n_name = 'EGYPT' )
                or
                ( n1.n_name = 'EGYPT' and n2.n_name = 'KENYA' )
            )
            and l_shipdate between to_date (date  '1995-01-01') and to_date (date '1996-12-31')
    ) shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year;
```

8. National Market Share Query (Q8)

This query determines how the market share of a given nation within a given region has changed over two years for a given part type.

Business Question

The market share for a given nation within a given region is defined as the fraction of the revenue, the sum of [l_extendedprice * (1 - l_discount)], from the products of a specified type in that region that was supplied by suppliers from the given nation. The query determines this for the years 1995 and 1996 presented in this year.

Functional Query Definition

```
select
    o_year,
    sum(case
        when nation = 'EGYPT' then volume
        else 0
    end) / sum(volume) as mkt_share
from
    (
        select
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
            and s_suppkey = l_suppkey
            and l_orderkey = o_orderkey
            and o_custkey = c_custkey
            and c_nationkey = n1.n_nationkey
            and n1.n_regionkey = r_regionkey
            and r_name = 'MIDDLE EAST'
            and s_nationkey = n2.n_nationkey
            and o_orderdate between to_date (date '1995-01-01') and to_date (date '1996-12-31')
            and p_type = 'MEDIUM BRUSHED NICKEL'
    ) all_nations
group by
    o_year
order by
    o_year;
```

9. Product Type Profit Measure Query (Q9)

This Query determines how much profit is made on a given line of parts, broken out by supplier nation and year.

Business Question

Product Type Profit Measure Query finds, for each nation and each year, the profit for all parts ordered in that year that contain a specified substring in their names and that were filled by a supplier in that nation. The profit is defined as the sum of [l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity] for all lineitems describing part in the specified line. The query lists the nations in ascending alphabetical order and, for each nation, the year and profit in descending order by year (most recent first).

Functional Query Definition

```
select
    nation,
    o_year,
    sum(amount) as sum_profit
from
    (
        select
            n_name as nation,
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
        from
            part,
            supplier,
            lineitem,
            partsupp,
            orders,
            nation
        where
            s_suppkey = l_suppkey
            and ps_suppkey = l_suppkey
            and ps_partkey = l_partkey
            and p_partkey = l_partkey
            and o_orderkey = l_orderkey
            and s_nationkey = n_nationkey
            and p_name like '%light%'
    ) profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
```

10. Returned Item Reporting Query (Q10)

The query identifies customers who might be having problems with the parts that are shipped to them.

Business Question

Returned Item Reporting Query finds the top 20 customers, in terms of their effect on lost revenue for a given quarter, who have returned part. The query considers only parts that were ordered in the specified quarter. The query lists the customer's name, address, nation, phone number, account balance, comment information and revenue lost. The customers are listed in descending order or lost revenue. Revenue lost is defined as sum(l_extendedprice * (1 - l_discount)) for all qualifying lineitems.

Functional Query Definition

```
select
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= to_date(date '1993-11-01')
    and o_orderdate < to_date(date '1993-11-01' + interval '3' month)
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
    and rownum < 21
group by
    c_custkey,
    c_name,
    c_acctbal,
    c_phone,
    n_name,
    c_address,
    c_comment
order by
    revenue desc;
```

11. Important Stock Identification Query (Q11)

This query finds the most important subset of suppliers' stock in a given nation.

Business Question

Important Stock Identification Query finds, from scanning the available stock of suppliers in a given nation, all the parts that represent a significant percentage of the total value of all available parts. The query displays the part number and the value of those part in descending order of value.

Functional Query Definition

```
select
    ps_partkey,
    sum(ps_supplycost * ps_availqty) as value
from
    partsupp,
    supplier,
    nation
where
    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
    and n_name = 'UNITED KINGDOM'
group by
    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost * ps_availqty) * 0.0001000000
            from
                partsupp,
                supplier,
                nation
            where
                ps_suppkey = s_suppkey
                and s_nationkey = n_nationkey
                and n_name = 'UNITED KINGDOM'
        )
order by
    value desc;
```

12. Shipping Modes and Order Priority Query (Q12)

This query determines whether selecting less expensive modes of shipping is negatively affecting the critical-priority orders by causing more parts to be received by customers after the committed date.

Business Question

Shipping Modes and Order Priority Query counts, by ship mode, for lineitems actually received by customers in a given year, the number of lineitems belonging to orders for which the l_receiptdate exceeds the l_commitdate for tow different specified ship modes. Only lineitems that were actually shipped before the l_commitdate are considered. The late lineitems are partitioned into tow groups, those with priority URGENT or HIGH, and those with a priority other than URGENT or HIGH.

Functional Query Definition

```
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
            or o_orderpriority = '2-HIGH'
            then 1
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
            and o_orderpriority <> '2-HIGH'
            then 1
        else 0
    end) as low_line_count
from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('MAIL', 'FOB')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= to_date (date '1997-01-01')
    and l_receiptdate < to_date (date '1997-01-01' + interval '1' year)
group by
    l_shipmode
order by
    l_shipmode;
```

13. Customer Distribution Query (Q13)

This query seeks relationships between customers and the size of their orders.

Business Question

Customer Distribution Query determines the distribution of customers by the number of orders they have made, including customers who have no record of orders, past or present. It counts and reports how many customers have no orders, how many have 1, 2, 3, etc. A check is made to ensure that the orders counted do not fall into one of several special categories of orders. Special categories are identified in the order comment column by looking for a particular pattern.

Functional Query Definition

```
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey) as c_count
        from
            customer, orders -- left outer join orders on
              where
                c_custkey(+) = o_custkey
                and o_comment not like '%unusual%deposits%'
        group by
            c_custkey
    ) -- c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;
```

14. Promotion Effect Query (Q14)

This query monitors the market response to a promotion such as TV advertisements or a special campaign.

Business Question

Promotion Effect Query determines what percentage of the revenue in a given year and month was derived from promotional parts. The query considers only parts actually shipped in that month and gives the percentage. Revenue is defined as (l_extendedprice * (1 - l_discount)).

Functional Query Definition

```
select
    100.00 * sum(case
        when p_type like 'PROMO%'
            then l_extendedprice * (1 - l_discount)
        else 0
    end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= to_date (date '1996-11-01')
    and l_shipdate < to_date (date '1996-11-01' + interval '1' month);
```

15. Top Supplier Query (Q15)

This query determines the top supplier so it can be rewarded, given more business, or identified for special recognition.

Business Question

Top Supplier Query finds the supplier who contributed the most to the overall revenue for parts shipped during a given quarter of a given year. In case of a tie, the query lists all suppliers whose contribution was equal to the maximum, presented in supplier number order.

Functional Query Definition

```
create view revenue0 (supplier_no, total_revenue) as
    select
        l_suppkey,
        sum(l_extendedprice * (1 - l_discount))
    from
        lineitem
    where
        l_shipdate >= to_date (date '1997-03-01')
        and l_shipdate < to_date (date '1997-03-01' + interval '3' month)
    group by
        l_suppkey;

-- 'c:\auxitools\out\appendix\stream\15.0'
select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue0
where
    s_suppkey = supplier_no
    and total_revenue = (
        select
            max(total_revenue)
        from
            revenue0
    )
order by
    s_suppkey;

drop view revenue0;
```

16. Part/Supplier Relationship Query (Q16)

This query finds out how many suppliers can supply parts with given attributes. It might be used, for example, to determine whether there is a sufficient number of suppliers for heavily ordered parts.

Business Question

Part/Supplier Relationship Query counts the number of suppliers who can supply parts that satisfy a particular customer's requirements. The customer is interested in parts of eight different sizes as long as they are not of a given type, not of a given brand, and not from a supplier who has had complaints registered at the Better Business Bureau. Results must be presented in descending count and ascending band, type, and size.

Functional Query Definition

```
select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#23'
    and p_type not like 'PROMO BURNISHED%'
    and p_size in (33, 9, 35, 38, 20, 13, 22, 14)
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
            s_comment like '%Customer%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;
```

17. Small-Quantity-Order Revenue Query (Q17)

This query determines how much average yearly revenue would be lost if orders were no longer filled for small quantities of certain parts. This may reduce overhead expenses by concentrating sales on larger shipments.

Business Question

Small-Quantity-Order Revenue Query considers parts of a given brand and with a given container type and determines the average lineitem quantity of such parts ordered for all orders (past and pending) in the 7-year database. What would be the average yearly gross (undiscounted) loss in revenue if orders for these parts with a quantity of less than 20% of this average were no longer taken?

Functional Query Definition

```
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#35'
    and p_container = 'JUMBO BOX'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
```

18. Large Volume Customer Query (Q18)

This query ranks customers based on their having placed a large quantity order. Large quantity orders are defined as those orders whose total quantity is above a certain level.

Business Question

Large Volume Customer Query fids a list of the top 100 customers who have ever placed large quantity orders. The query lists customer name, customer key, the order key, date and total price and the quantity for the order.

Functional Query Definition

```
select
        c_name,
        c_custkey,
        o_orderkey,
        o_orderdate,
        o_totalprice,
        sum(l_quantity)
from
        customer,
        orders,
        lineitem
where
        o_orderkey in (
                select
                        l_orderkey
                from
                        lineitem
                group by
                        l_orderkey having
                                sum(l_quantity) > 315
        )
        and c_custkey = o_custkey
        and o_orderkey = l_orderkey
        and rownum < 101
group by
        c_name,
        c_custkey,
        o_orderkey,
        o_orderdate,
        o_totalprice
order by
        o_totalprice desc,
        o_orderdate;
```

19. Discounted Revenue Query (Q19)

Discounted Revenue Query reports the gross discounted revenue attributed to the sale of selected parts handled in a particular manner. This query is an example of code such as might be produced programmatically be a data mining tool.

Business Question

Discounted Revenue Query finds the gross discounted revenue for all orders for three different types of parts that were shipped by air or delivered in person. Parts are selected based on the combination of specific brands, a list of container, and a range of sizes.

Functional Query Definition

```
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#41'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 5 and l_quantity <= 5 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#45'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 13 and l_quantity <= 13 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#22'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 20 and l_quantity <= 20 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );
```

20. Potential Part Promotion Query (Q20)

Potential Part Promotion Query identifies suppliers in a particular nation having selected parts that may be candidates for a promotional offer.

Business Question

Potential Part Promotion Query identifies suppliers who have an excess of a given part available; an excess is defined to be more than 50% of the parts like the given part that the supplier shipped in a given year for a given nation. Only parts whose names share a certain naming convention are considered.

Functional Query Definition

```
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like 'cornflower%'
            )
            and ps_availqty > (
                select
                    0.5 * sum(l_quantity)
                from
                    lineitem
                where
                    l_partkey = ps_partkey
                    and l_suppkey = ps_suppkey
                    and l_shipdate >= to_date (date '1996-01-01')
                    and l_shipdate < to_date (date '1996-01-01' + interval '1' year)
            )
    )
    and s_nationkey = n_nationkey
    and n_name = 'VIETNAM'
order by
    s_name;
```

21. Suppliers Who Kept Orders Waiting Query (Q21)

This query identifies certain suppliers who were not able to ship required parts in a timely manner.

Business Question

Suppliers Who Kept Orders Waiting Query identifies suppliers, for a given nation, whose product was part of a multi-supplier order (with current status of 'F') where they were the only supplier who failed to meet the committed delivery date.

Functional Query Definition

```
select
      s_name,
      count(*) as numwait
from
      supplier,
      lineitem l1,
      orders,
      nation
where
      s_suppkey = l1.l_suppkey
      and o_orderkey = l1.l_orderkey
      and o_orderstatus = 'F'
      and l1.l_receiptdate > l1.l_commitdate
      and exists (
            select
                  *
            from
                  lineitem l2
            where
                  l2.l_orderkey = l1.l_orderkey
                  and l2.l_suppkey <> l1.l_suppkey
      )
      and not exists (
            select
                  *
            from
                  lineitem l3
            where
                  l3.l_orderkey = l1.l_orderkey
                  and l3.l_suppkey <> l1.l_suppkey
                  and l3.l_receiptdate > l3.l_commitdate
      )
      and s_nationkey = n_nationkey
      and n_name = 'PERU'
      and rownum < 101
group by
      s_name
order by
      numwait desc,
      s_name;
```

22. Global Sales Opportunity Query (Q22)

Global Sales Opportunity Query identifies geographies where there are customers who may be likely to make a purchase.

Business Question

Global Sales Opportunity Query counts how many customers within a specific range of country codes have not placed orders for 7 years but who have a greater than average "positive" account balance. It also reflects the magnitude of that balance. Country code is defined as the first two characters of c_phone.

Functional Query Definition

```
select
    cntrycode,
    count(*) as numcust,
    sum(c_acctbal) as totacctbal
from
    (
        select
            substr(c_phone, 1, 2) as cntrycode,
            c_acctbal
        from
            customer
        where
            substr(c_phone, 1, 2) in
                ('15', '19', '16', '20', '14', '22', '10')
            and c_acctbal > (
                select
                    avg(c_acctbal)
                from
                    customer
                where
                    c_acctbal > 0.00
                    and substr(c_phone, 1, 2) in
                        ('15', '19', '16', '20', '14', '22', '10')
            )
            and not exists (
                select
                    *
                from
                    orders
                where
                    o_custkey = c_custkey
            )
    ) custsale
group by
    cntrycode
order by
    cntrycode;
```

Glossary

TPC: The TPC is a non-profit corporation founded to define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry.

TPC-H : TPC-H is an ad-hoc, decision support benchmark.

RF: Refresh Function.

DBMS: Database Management System.

RDBMS: Relational Database Management System.

DBA: Database Administrator.

DBCA: Database Configuration Assistant.

GUI: graphical user interface.

SQL: Structure Query Language.

OLTP: Online Transaction Processing.

OLAP: Online Analytical Processing.

OS: Operating System.

RAM: Random Access Memory.