UNCLASSIFIED

AFIT/EN-TR-93-5

Air Force Institute of Technology

"Magic mirror on the wall, who's the fastest Database of them all?" A Survey of Database Benchmarks

Timothy J. Halloran Capt, USAF
Mark A. Roth Maj, USAF

21 June 1993

"Magic mirror on the wall, who's the fastest Database of them all?" A Survey of Database Benchmarks

Timothy J. Halloran
Mark A. Roth[‡]

Abstract

Benchmarks are important tools for measuring the performance of database management systems (DBMS) and for understanding vendor claims of performance. This paper defines DBMS benchmarks, explores the role of the Transaction Processing Performance Council as the only benchmark standards organization, and surveys eight existing DBMS benchmarks for on-line transaction processing, relational, and object-oriented databases.

1 Introduction

When examining the performance of a commercial database management system (DBMS), one is bombarded with vendor performance claims. Each vendor will shower you with claims of "tpsA-Local" ratings, or top performance on the "TPC-C" benchmark. If you are looking at a commercial object-oriented DBMS, vendors will tell you they have the best performance on the "Cattell" (or OO1 or "Sun") benchmark. The use of benchmark performance measurements by commercial DBMS vendors seems a bit like the Queen in *Snow White* asking the magic mirror who is the fairest. If the answer is not to their liking, they go to great pains to change the answer. Vendors engage in "Benchmark Wars" and "Benchmarketing"[5]. The best defense is to have knowledge about the DBMS benchmarks used today.

DBMS benchmarks are a way to measure the performance and/or functionality of a DBMS. They can also be used to find the lowest cost DBMS and computer system for a required job. This paper surveys DBMS benchmarks. First, the criteria for a good DBMS benchmark is covered. Second, the role of the only DBMS benchmark standards organization, the Transaction Processing Performance Council, is examined. Then eight important DBMS benchmarks are looked at. For each benchmark we point out the important strengths and weaknesses of the benchmark. Included are benchmarks for on line transaction processing (OLTP), relational, and object-oriented DBMSs. The following items about each benchmark are examined:

- The benchmark problem domain
- The benchmark database
- The benchmark operations
- The measurements (or results) of the benchmark

For more detailed information about any specific benchmark, the source documents on that benchmark should be examined.

2 DBMS Benchmarks

DBMS benchmarks are *domain-specific benchmarks*. These benchmarks attempt to quantitatively measure the performance of a DBMS in a specific domain area, such as decision support or OLTP. Gray in [5] proposes the following criteria for a good domain-specific benchmark:

- Relevant
- Portable
- Scalable
- Simple

A benchmark must be *relevant* to be a useful yardstick for DBMS performance. For example, if you are planning to use a DBMS for OLTP, then results of OLTP benchmarks can aid in selection of a DBMS and a computer

[‡]The authors are with the Department of Electrical and Computer Engineering (AFIT/ENG), Air Force Institute of Technology, 2950 P ST, Wright-Patterson AFB, OH 45433-7765.

system. But, if you are planning to use the database for a decision support system, then OLTP benchmarks are not useful, because they are not relevant to the decision support domain.

A benchmark must be *portable* so that it can be run on several different DBMSs and computer systems. Ideally, a benchmark should be able to be run on all the DBMSs which support the domain (e.g., OLTP) for which it measures performance.

A benchmark should be *scalable* to large and small computer systems. As the capabilities of the computer system increase, the benchmark should "scale-up" to credibly measure the performance of that computer system. Gray also notes that a benchmark should scale up to new types of computer systems (i.e. parallel computer systems) as "computer performance and architecture evolve."[5:5]

A benchmark should be *simple* so it can be understood and easily implemented. If a benchmark is as complex as your intended application, then there would be little point to using the benchmark (your application could be used to measure DBMS performance). A benchmark must be a small and simple program which can be used as a yardstick to evaluate a system under test (a DBMS and computer system).

DBMS benchmarks are not perfect, and can be abused by vendors. Gray sites two major benchmark abuses: "Benchmark Wars", and "Benchmarketing"[5]. The "Benchmark Wars" occur between DBMS vendors trying to maintain top performance on a specific benchmark. If one vendor loses to another, the losing vendor reruns the benchmark with better "gurus". If the vendor succeeds in getting better results, the other vendor does the same thing. This can continue to the point were modifications are being made to the DBMS software specifically to make the benchmark run faster. "Benchmarketing" is where a benchmark is modified (or a new benchmark is created) to make a specific DBMS product do extremely well.

3 DBMS Benchmark Standards Organizations

There is only one existing standards body for DBMS benchmarks: The Transaction Processing Performance Council(TPC). The TPC is a non-profit corporation founded in August 1988. The mission of the TPC is "to define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry."[9:1] The TPC was created because of the lack of agreed upon benchmarks for measuring DBMS performance. The TPC performs the following services:

- *Defines Standard Benchmarks* — The TPC has created three standard benchmarks to date: TPC-A, TPC-B, and TPC-C (which will be examined in this paper), and has two more in the works: TPC-D (decision support domain) and TPC-E (enterprise domain).

- *Full Disclosure of Results* — All companies which claim a performance measure on a TPC benchmark must submit a detailed report to the TPC (called a full disclosure report). This report documents the benchmark's compliance with the TPC benchmark standard.

- *Quarterly Report* — The TPC publishes a quarterly report which contains summaries of all the benchmark results published that quarter.

The TPC has 41 current members which include both DBMS software and computer hardware vendors*.

4 Benchmarks for OLTP and Relational DBMSs

The following four benchmarks for OLTP and relational DBMSs will be examined:

- TPC Benchmark A (TPC-A)
- TPC Benchmark B (TPC-B)
- TPC Benchmark C (TPC-C)
- Wisconsin Benchmark

The first three are the standard benchmarks defined by the TPC. The Wisconsin benchmark is a benchmark for complex relational queries. TPC-A and TPC-B will be covered together, because they are very similar. They will be covered in detail. TPC-C and the Wisconsin benchmark will not be covered in as much detail due to their complexity.

4.1 TPC-A and TPC-B

The TPC-A and TPC-B benchmarks are very similar, and will be examined together. The TPC-A benchmark was developed in 1989 by the Transaction Processing Performance Council, TPC-B was developed in 1990. TPC-A and TPC-B use the same database and transaction profile, ACID† requirements, and costing formula. The major

*The Transaction Processing Performance Council(TPC) can be reached at (408)295-8894.

†Atomicity, Consistency, Isolation (or serializability), and Durability

difference between the two benchmarks is that TPC-B allows the use of transaction generation processes to create transactions, while TPC-A requires the use of terminal emulation to create transactions. The TPC-A benchmark is a simple OLTP benchmark, while the TPC-B benchmark may be thought of as a database stress test.

The specifications for TPC-A and TPC-B state they "exercise the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing update-intensive database services."[5] Both TPC-A and TPC-B are defined in terms of a banking application. The bank has one or more branches, and each branch has multiple tellers (each with a terminal to the database). All the bank customers have an account. The final metric from the TPC-A and TPC-B benchmarks is throughput as measured in transactions per second.

4.1.1 Benchmark Database

The database consists of four tables (or files): Account, Branch, Teller, and History. The relationships between these tables is shown in Figure 1. Figure 1 is an entity/relationship diagram for the database.

The Account table contains the following fields:

- *Account_ID* (The key for the table)
- *Branch_ID* (The branch where the account is held)
- *Account_Balance*

The Branch table contains the following fields:

- *Branch_ID* (The key for the table)
- *Branch_Balance*

The Teller table contains the following fields:

- *Teller_ID* (The key for the table)
- *Branch_ID* (The branch where the teller is located)
- *Teller_Balance*

The History table contains the following fields:

- *Account_ID* (Updated by transaction)
- *Teller_ID* (Performed the transaction)
- *Branch_ID* (Associated with teller)
- *Amount*
- *Time_Stamp* (Time of the transaction)

```
BEGIN TRANSACTION
  Update Account where Account_ID = Aid:
    Read Account_Balance from Account
    Set Account_Balance = Account_Balance + Delta
    Write Account_Balance to Account
  Write to History:
    Aid, Tid, Bid, Delta, Time_stamp
  Update Teller where Teller_ID = Tid:
    Set Teller_Balance = Teller_Balance + Delta
    Write Teller_Balance to Teller
  Update Branch where Branch_ID = Bid:
    Set Branch_Balance = Branch_Balance + Delta
    Write Branch_Balance to Branch
COMMIT TRANSACTION
```

Figure 2: TPC-A and TPC-B Transaction Profile

The benchmark specification requires that all branches must have the same number of tellers, and that all branches must have the same number of accounts. The number of rows in each table is not a fixed value. It is scaled based upon the throughput rate for which the test is configured.

4.1.2 Benchmark Operations

Only one transaction is performed on the benchmark database. The transaction profile is shown in Figure 2. Aid (Account_ID), Tid (Teller_ID), and Bid (Branch_ID) are keys. For TPC-A, the Aid, Tid, Bid, Delta are read from a teller terminal, the transaction is processed, then Aid, Tid, Bid, Delta, and Account_Balance are written back to the terminal. For TPC-B, the Aid, Tid, Bid, and Delta are provided by a driver, and only Account_Balance is returned to the driver after the transaction has been processed. It is important to realize that the TPC-A benchmark measures the time it takes messages to pass through the communication network to and from the teller terminals, and TPC-B does not.

4.1.3 Benchmark Measurements

TPC-A and TPC-B provide two important metrics: a tps and a K$/tps. The tps is a throughput measurement which stands for "transactions per second". To avoid confusion with other (older) similar benchmarks which create a tps metric (i.e., DebitCredit and TP1[1]), the TPC-A and TPC-B benchmarks prefix the tps metrics. TPC-A uses "tpsA-Local" and "tpsA-Wide". "tpsA-Local" states the test was run using a local communications network, and "tpsA-Wide" states the test was run using a wide

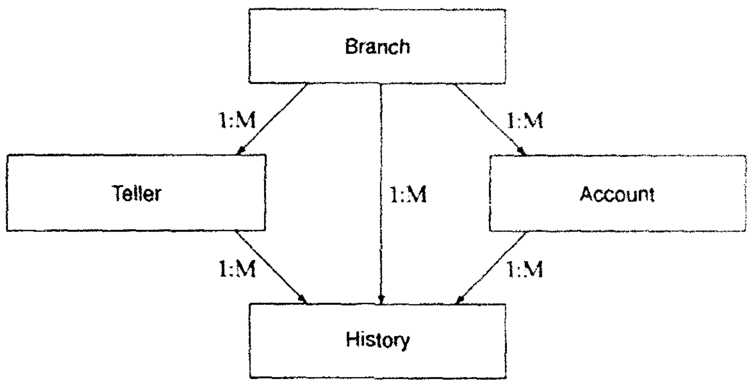

Figure 1: TPC-A and TPC-B Entity/Relationship Diagram

area communications network. TPC-B uses "tpsB" for its results.

What is required to generate a valid tps rating for TPC-A and TPC-B is not obvious, and requires some explanation. The basic calculation is simple: to obtain a tps rating, the number of transactions which started and completed during the test interval is divided by the elapsed time of the test. But, in order for the tps metric to be a valid for the TPC-A or TPC-B benchmark, several requirements must be met. These are:

- The database table sizes must be scaled properly
- The test interval must be at least 15 minutes (and no longer than an hour)
- 90% of the transactions must have less than a 2 second response time (to the terminal for TPC-A, to the driver for TPC-B)
- Each terminal (for TPC-A) creates a new transaction (on average) every 10 seconds

First, the database table sizes must be scaled to the throughput goal of the test. TPC-A and TPC-B are scaled based upon DBMS throughput. If a benchmark test is trying to measure a throughput of 10 tps, then the database size must be scaled for that level of throughput. A tps throughput measurement is only allowed to be as high as the database table size allows. For each tps configured, the benchmark specification states that the database tables must have the following number of rows:

Table	Number of Rows
Account	100,000 rows
Teller	10 rows
Branch	1 row

In addition to the required table sizes, for TPC-A there must be 10 terminals for each tps configured.

Second, the test must be run in a steady state for at least a time of 15 minutes and no longer than one hour, but the test system must have enough resources to run the test for a total of 8 hours.

Third, 90% of all transactions during the test must have a response time of under 2 seconds.

Example 1: Consider a TPC-A test system configured for 10 tps using a local area network. To allow 10 tps the test database and computer system must use a minimum of:

Item	Number
Account rows	1,000,000
Teller rows	100
Branch rows	10
Terminals	100

The test on the system is run for 20 minutes, and the 90th percentile response time is 1.85 (below the 2 second requirement). During the 20 minutes suppose 11,261 transactions are started and committed. The tps rating would be 9.38 tpsA-Local ($\frac{11,261}{20\cdot 60}$). Since this value is below 10 it is valid tps rating for TPC-A. But if 13,204 transactions started and committed during the test, the tps rating of 11 tps would be invalid. This is because 11 tps is larger than the 10 tps throughput for which the test system was configured.□

The second measurement from TPC-A and TPC-B is the K$/tps. This value is obtained by dividing the cost of the system by the measured tps rating. The benchmark standard is very specific about what items are to be included in the cost of the computer system. It includes cost of the computer hardware, terminals, communication lines, database software, and maintenance.

Example 2: In the TPC-A test above (example 1), assume that the system under test costs \$140,000. The system had a throughput metric of 9.38 tpsA-Local. The K\$/tps metric would be 14.9 K\$/tps ($\frac{140}{9.38}$).□

These two benchmarks are widely used by DBMS vendors today. And TPC-A and TPC-B summary results are regularly published in computer industry literature.

A major strength of the TPC-A and TPC-B benchmarks is their simplicity. These benchmarks produce very simple results (tps measurements). Because of these simple results it is important to recognize the limitations of the benchmarks. TPC-A is a useful yardstick for simple OLTP performance capabilities, and TPC-B provides a simple DBMS stress test, but because of the simple transaction used in both benchmarks, they are of absolutely no value for measuring how well a DBMS will perform on complex queries.

4.2 TPC-C

The TPC-C benchmark was developed in 1992 by the Transaction Processing Performance Council. This benchmark was designed to simulate an OLTP workload. It, like the TPC-A benchmark, is a useful yardstick for OLTP performance capabilities. The TPC-C benchmark is much more complex than the TPC-A and TPC-B benchmarks (it requires 111 pages for its specification, while TPC-A and TPC-B require 39 pages and 38 pages respectively). TPC-C simulates a business where terminal operators execute transactions against a database. The TPC-C benchmark is specified to exercise the following components of an OLTP database system[8]:

- The simultaneous execution of multiple transaction types that span a breadth of complexity.
- On-line and deferred transaction execution modes
- Multiple on-line terminal sessions
- Moderate system and application execution time
- Significant disk input/output
- Transaction integrity (ACID properties)
- Non-uniform distribution of data access through primary and secondary keys
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Contention on data access and update

4.2.1 Benchmark Database

The TPC-C benchmark database represents a wholesale supplier with several sales districts. The supplier has warehouses which cover a group of sales districts. Each sales district has a group of customers. For the TPC-C benchmark the following rules are specified for the benchmark database:

- Each regional warehouse covers 10 districts
- Each district serves 3,000 customers
- All warehouses maintain stocks for the 100,000 items sold by the company
- The database size is scaled by adding more warehouses (all the other cardinalities are fixed)

The benchmark database size is scaled based upon the throughput of the DBMS (like TPC-A and TPC-B).

4.2.2 Benchmark Operations

The TPC-C benchmark operations are based around the types of transactions which would be typical in an order-entry environment. The following transactions are run on the TPC-C database:

1. *New-Order Transaction*: This transaction enters a complete order in a single database transaction.
2. *Payment Transaction*: This transaction updates a customers balance and reflects the payment on district and warehouse sales statistics.
3. *Order-Status Transaction*: This transaction queries the status of a customer's last order.
4. *Delivery Transaction*: This transaction processes 10 new orders (the orders are delivered).
5. *Stock-Level Transaction*: This transaction determines the number of items that have a stock level below a threshold level.

All of these transactions are executed during the TPC-C benchmark. They are done in the frequency which would be expected in a real business.

4.2.3 Benchmark Measurements

The final metric from the TPC-C benchmark is throughput in transactions per minute. The metric is called "tpmC". As in TPC-A and TPC-B, the reported throughput may not exceed the maximum allowed by the database size.

The complexity of the TPC-C benchmark is both a strength and a weakness. It is a strength, because the benchmark is more realistic (for an OLTP application), and a weakness because it makes the results of the benchmark more difficult to interpret. As with all the TPC benchmarks, the standardization of the benchmark is a strength. The benchmark leaves little flexibility in implementation (so it is less likely to be abused by vendors). A weakness of this benchmark is the single throughput (tpmC) which is generally reported (in summaries of results), but more detailed information is required in the full disclosure report.

4.3 The Wisconsin Benchmark

The Wisconsin Benchmark was developed by Bitton, DeWitt, and Turbyfill in 1983[5]. This benchmark measures DBMS performance on a variety of complex relational queries. 32 queries are done on the benchmark database, each query attempts to measure DBMS performance on one, or a group of, basic relational operators (i.e. selection, projection, or join).

4.3.1 Benchmark Database

The benchmark database consists of three tables. The first table contains 1,000 tuples and is named `ONEKTUP`. The other two tables contain 10,000 tuples each and are named `TENKTUP1` and `TENKTUP2`. The fields in the tables are all the same, and are synthetically generated relations. DeWitt states that this choice was made to make the database scalable, and to "permit systematic benchmarking."[5:122]

4.3.2 Benchmark Operations

The benchmark measures performance on the following types of queries:

1. *Selection Queries*
2. *Join Queries*
3. *Projection Queries*
4. *Aggregate Queries*
5. *Update Queries*

There are a total of 32 queries in the benchmark specification.

4.3.3 Benchmark Measurements

For each of the 32 queries in the benchmark, elapsed time is used as the performance metric. This is the wall clock time from when the query was started until it completes.

This benchmark had a major impact on commercial DBMSs when it was created (1983). As DeWitt states, "by pointing out the performance warts of each system, vendors were forced to significantly improve their systems in order to remain competitive."[5:120] For example, at the time the benchmark was released, nested loops was the only join method provided by the ORACLE and IDM 500 DBMSs[5]. DeWitt reports that "each required over five hours to execute" one of the benchmark join queries[5:134].

The Wisconsin benchmark currently is being used to evaluate the performance of database systems running on parallel processors.[5]

The major strength of this benchmark is its focus on query performance. The problem with this is that one has to have to have some education in relational database theory to understand the results. If a user doesn't understand how a selection query is different from a join query, the results from this benchmark will not be useful. But for domains, such as decision support, where complex queries are necessary, this benchmark can be helpful in evaluating the performance of a DBMS.

5 Benchmarks for Object-Oriented DBMSs

Though performance is important to most applications which could use object-oriented DBMSs, Cattell maintains that little work has been done in the area[3]. Only the following four benchmarks have been proposed for object-oriented DBMSs (none of which are TPC standards):

- Simple Database Operations Benchmark
- Object Operations Version 1 (OO1) Benchmark
- HyperModel Benchmark
- OO7

The Simple Database Operations benchmark was developed first. The HyperModel and OO1 benchmarks are both based on the Simple Database Operations benchmark, but HyperModel is a more complex benchmark than OO1. The OO7 benchmark is a new benchmark created at the University of Wisconsin (the creators of the Wisconsin Benchmark). Figure 3 shows the evolution of object-oriented DBMS benchmarks. Each of these benchmarks will be examined.

5.1 Simple Database Operations Benchmark

This benchmark was proposed in 1987 by Rubenstein, Kubicar, and Cattell[7]. They created the benchmark because existing relational DBMS benchmarks were poor measures for the applications they were working on. They needed a measure of "*response time* for simple queries" [7:387].

As an example, consider drawing a polygon on a computer screen where the lines which make up the polygon are stored in the DBMS. The program would start by querying the database for the first line in the polygon. When that line was returned, it would be drawn on the screen. This process would be repeated for each line in the polygon. In a complex CAD drawing there could be hundreds of thousands of lines. This is the type of application Rubenstein, Kubicar, and Cattell were interested in providing a benchmark for, but they used an easier to understand database of documents and authors rather than polygons for their benchmark.

5.1.1 Benchmark Database

The benchmark uses a database of *document* and *person* records. Documents are related to people by a relationship called *author*. 5,000 documents, 20,000 persons, and 15,000 author relationships are created in the benchmark database. To allow the benchmark to "scale up" to larger databases, the benchmark proposes the same measurements also be run on a database ten times and one hundred times larger [7:389].

5.1.2 Benchmark Operations

For each different size database, the performance of the following seven different operations is measured:

1. *Name Lookup*: Find the name of a single person.
2. *Range Lookup*: Find the names of people with birth dates in a particular 10-day period.
3. *Group Lookup*: Given a random document, find all authors for that document.
4. *Reference Lookup*: Find the name and birth date for a single author of a random document.
5. *Record Insert*: Create a new author record, and add it to the database.
6. *Sequential Scan*: Retrieve, one at a time, the title of every document in the database.
7. *Database Open*: Perform all the operations necessary to make the DBMS available to run an application program.

5.1.3 Benchmark Measurements

For each of the benchmark operations the performance measurement is the response time of the operation[7]. The response time is the elapsed time from when the operation is started until it completes.

The scaling of the benchmark database in this benchmark is limited (only three sizes), this is a weakness of this benchmark (and most object-oriented DBMS benchmarks). Most of these benchmarks intend to measure performance of the object-oriented DBMS when the entire database can fit in memory, and then when it can not all fit in memory. The main strength of this benchmark is that it is quite simple, but it has not turned out to be very popular, and has been overshadowed by the OO1 benchmark.

5.2 HyperModel Benchmark

This benchmark was proposed in 1990 by Berre and Anderson [6:75–91]. The HyperModel benchmark is a very complex benchmark for object-oriented DBMSs, because it measures a large number of different operations. The creators of the HyperModel benchmark concluded that the Simple Database Operations benchmark did not measure enough database operations on a complex enough database to be representative of a wide variety of engineering applications [6].

5.2.1 Benchmark Database

The HyperModel benchmark uses a database which represents hypertext. Hypertext consists of nodes and links. Nodes contain information such as text, graphics, or sound. Links maintain relationships between pieces of information. Berre and Anderson state that "Hypertext has been proposed as a good model for use in Computer Aided Software Engineering (CASE) because it is possible to store software and documentation as hypertext."[6:75]

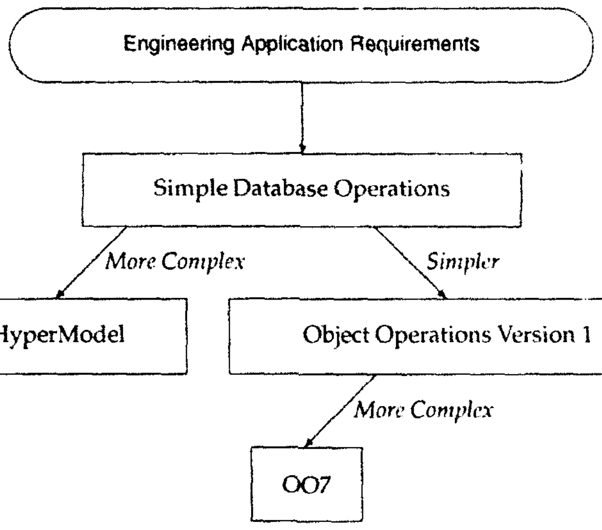
Engineering Application Requirements
Simple Database Operations
More Complex
Simpler
HyperModel
Object Operations Version 1
More Complex
OO7

Figure 3: The Evolution of Object-Oriented DBMS Benchmarks

5.2.2 Benchmark Operations

The following operations are measured in the HyperModel benchmark:

1. *Name Lookup*: A lookup is performed on a hypertext node.
2. *Range Lookup*: A range of hypertext nodes is looked up.
3. *Group Lookup and Reference Lookup*: Same as the Simple Database Operations benchmark, but it is extended to one-to-many, many-to-many, and many-to-many with attribute relationships as well.
4. *Sequential Scan*: Same as the Simple Database Operations benchmark, the entire database is retrieved.
5. *Closure Traversal*: Starting at a random hypertext node, find all the nodes transitively reachable by a relationship. This is done for all the types of relationships in the database.
6. *Closure Operations*: The same as closure traversal, but an operation will be performed at each node found during the traversal.
7. *Editing*: This operation changes the text found at a hypertext node, then changes it back to its previous value. The operation is also done for a hypertext node with a graphics image (picture) stored in it.
8. *Create and Delete*: This operation creates a node, and then deletes it.
9. *Open and Close*: Same as Simple Database Operations benchmark, but database close time is also measured.

5.2.3 Benchmark Measurements

The performance measurement is the elapsed time of each benchmark operation. The HyperModel benchmark specifies that each operation must be run 50 times, and that the database must be shutdown and restarted before each new type of operation is started [6].

The realistic database used in the benchmark and the realism gained by the complexity of the benchmark operations are this benchmarks greatest strengths. The benchmark's complexity is also a weakness. The complexity makes the benchmark difficult to implement, and the results difficult to understand. This benchmark has not proved to be very popular with object-oriented DBMS vendors.

5.3 Object Operations 1

This benchmark was proposed in 1991 by Cattell and Skeen of Sun Microsystems [4]. It is simpler than the Simple Database Operations benchmark (which Cattell also worked on), and is much simpler than the HyperModel benchmark. Cattell and Skeen admit the benchmark is representative of a smaller group of engineering applications than the HyperModel benchmark, but state they were trying to create a "generic benchmark" [4:2–3]. This has some merit, because the OO1 benchmark is much simpler to implement than the HyperModel benchmark. The OO1 benchmark is also known as the "Cattell" or "Sun" benchmark.

5.3.1 Benchmark Database

The database used for OO1 consists of connected parts. A connection goes from one part object to another part object, and a single part object may have several to and from connections. For each part, three connections to other parts are created. These connections must ensure "locality of reference" by connecting parts to parts which are closest to them (Part-id numbers which are numerically close are defined to be close together)[4:4–5].

OO1 measures performance on two different sized databases, called small and large. The small database consists of 20,000 parts and 60,000 connections. The large database is ten times larger than the small, hence having 200,000 parts and 600,000 connections. The authors of OO1 intended that the small database would fit in the database management system's memory buffer (or working set), while the large database would not fit in the memory buffer. They state that fitting in the working set is the "most important distinction" between the small and large databases [4:5–6].

5.3.2 Benchmark Operations

The OO1 benchmark measures are as follows:

1. *Lookup*: Lookup 1,000 (10,000 for the large database) parts in the database.

2. *Traversal*: Pick a single part, then find all parts connected to it (directly or indirectly) up to seven levels deep.

3. *Insert*: Create 100 (1,000 for the large database) new parts with three connections per part.

During each of the measurements it is required that a null procedure (representing some work done in an application program) in a programming language be called at each step. This requirement makes the benchmark "interactive" [4:7].

The benchmark forbids any of the operations being done as a single database call, which is how a relational DBMS might perform the operations.

A unique feature of the OO1 benchmark is that it must be run remotely. This means that the database must reside on one computer (server), and the benchmark application (client) must reside on another. The two computers are connected via a network. Figure 4 shows this configuration. The authors state that a remote database configuration is "the most realistic representation of engineering and office databases" [4:5].

The three OO1 benchmark measurements are run ten times. The results of the first run are the "cold start results," and the "asymptotic best times" on the remaining runs are the "warm start results" [4:8].

The OO1 benchmark has been very popular with object-oriented DBMS vendors (it has become a de-facto standard), probably in large part due to its simplicity. This benchmark's simplicity, and its attempt to measure the effectiveness of client caching are its strengths. Its poor coverage of the performance of a large number of the functional elements required in an object-oriented DBMS is its major weakness.

5.4 OO7

The OO7 benchmark was proposed by Carey, DeWitt, and Naughton in 1993. The work to develop this benchmark was done at the University of Wisconsin-Madison. The OO7 benchmark is proposed as a "comprehensive" performance profile of an object-oriented DBMS.[2:1] One of the interesting features of the OO7 benchmark is that it evaluates the performance of the query processor of the object-oriented DBMS (if it has one). The OO7 benchmark is more complex than the OO1 benchmark, but it is more focused than the HyperModel benchmark. The OO7 benchmark produces a set of numbers as its output metrics (not a single metric).

5.4.1 Benchmark Database

The benchmark database used for the OO7 benchmark is very complex. The database consists of a complex object hierarchy. The levels of the hierarchy (from bottom to top) are listed below:

1. *Atomic Parts*

2. *Composite Parts* composed of atomic parts, with associated documentation objects

3. *Base Assemblies* composed of composite parts

4. *Complex Assemblies* composed of base assemblies

5. *Design Objects* composed of complex assemblies

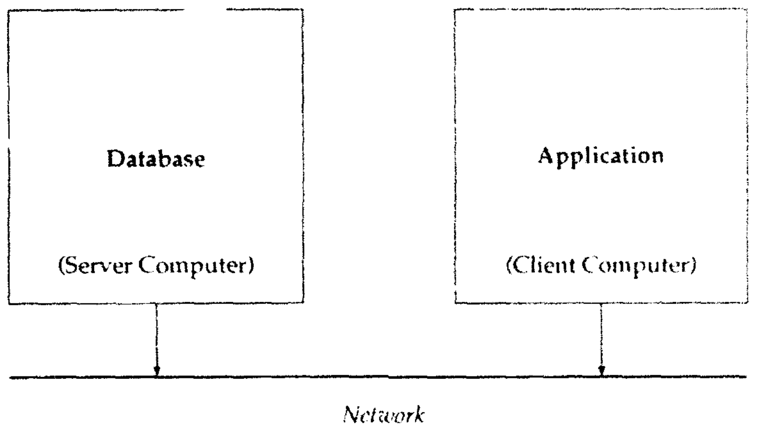

Figure 4: Remote or Client-Server Database Configuration

6. *Modules* composed of design objects, with associated documentation objects

Each object class in the database has several discrete attributes, and connections between classes are set up in the database. The OO7 benchmark scales the database to three different sizes: small, medium, and large.

5.4.2 Benchmark Operations

The benchmark measures performance on the following types of operations:

1. Traversals

2. Queries

3. Structural Modification Operations

5.4.3 Benchmark Measurements

The performance measurement is the elapsed time (or response time) of each benchmark operation.

This benchmark is very new, and it remains to be seen if it will become popular (specific parts of the benchmark may become popular with vendors, if their product performs well on them). The benchmark is very complex, and the results will probably have to be accompanied with a description of the benchmark operations (which was done in [2]). The strength of this benchmark is that it is very comprehensive in its measurements. This benchmark measures performance on a wide spectrum of object-oriented DBMS functionality.

6 Conclusion

This paper has reviewed eight benchmarks which are used for the performance measurement of DBMSs. For each benchmark the database, operations, and results used by the database have been discussed. Table 1 summarizes the benchmarks covered in this paper. DBMS benchmarks can be a useful tool for evaluating commercial DBMSs, but they can also be abused. A good understanding of DBMS benchmarks can help one to know when and when not to use a benchmark.

References

[1] Anon, et al. "A Measure of Transaction Processing Power," *Datamation*, *31*(7):112–118 (April 1985).

[2] Carey, Michael J., et al. *The OO7 Benchmark*. Technical Report, University of Wisconsin-Madison, April 12, 1993. available via anonymous ftp from cs.wisc.edu.

[3] Cattell, R.G.G. *Object Data Management: Object-Oriented and Extended Relational Database Systems*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1991.

[4] Cattell, R.G.G. and J. Skeen. "Object Operations Benchmark," *ACM Transactions on Database Systems*, *17*(1):1–31 (March 1992).

[5] Gray, Jim, editor. *The Benchmark Handbook For Database and Transaction Processing Systems*. San Mateo, California: Morgan Kaufmann Publishers, Inc., 1991.

Table 1: DBMS Benchmark Summary

Benchmark	Domain	Simplicity	TPC Standard
TPC-A	Simple OLTP	Simple	Yes
TPC-B	DBMS Stress Test	Simple	Yes
TPC-C	Complex OLTP	Complex	Yes
Wisconsin	Query Performance	Complex	No
Simple Database Operations	Object Operations	Simple	No
HyperModel	Object Operations	Complex	No
OO1	Object Operations	Simple	No
OO7	Object Operations	Complex	No

[6] Gupta, Rajiv and Ellis Horowitz, editors. *Object-Oriented Databases with Applications to CASE, Networks, and VLSI CAD.* Englewood Cliffs, NJ 07632: Prentice Hall, 1991.

[7] Rubenstein, W.B., et al. "Benchmarking Simple Database Operations," *Proceedings ACM SIGMOD*, 387–394 (May 1987).

[8] Transaction Processing Performance Council, "TPC Benchmark C Standard Specification," 13 August 1992.

[9] Transaction Processing Performance Council, "TPC Press Backgrounder," 1992.

21 June 1993 Technical Report

"Magic mirror on the wall, who's the fastest Database of them all?"
A Survey of Database Benchmarks

Timothy J. Halloran, Capt, USAF
Mark A. Roth, Maj, USAF

Air Force Institute of Technology, WPAFB OH 45433-7765

AFIT/EN-TR-93-5

Distribution Unlimited

Benchmarks are important tools for measuring the performance of database management systems (DBMS) and for understanding vendor claims of performance. This paper defines DBMS benchmarks, explores the role of the Transaction Processing Performance Council as the only benchmark standards organization, and surveys eight existing DBMS benchmarks for on-line transaction processing, relational, and object-oriented databases.

Database Benchmarks, TPC 12

UNCLASSIFIED UNCLASSIFIED UNCLASSIFIED UL