THE NEAL NELSON DATABASE BENCHMARK™:

A BENCHMARK BASED ON THE REALITIES OF BUSINESS

Neal Nelson & Associates is an independent benchmarking firm based in Chicago. They create and market benchmarks as well as offering consulting services on computer performance topics. This chapter describes the Neal Nelson Database Benchmark™. The chapter is divided into the following sections:

1) Introduction
2) Description of the benchmark operation and methodology
3) Description of the database design and organization
4) Sample queries
5) Sample test results
6) Observations and findings from customer uses of the benchmark
7) Checklist to evaluate the validity of database benchmark data
8) Conclusion

1. Introduction

Database Management Systems are commonly benchmarked by several different types of organizations. Each of these organizations has a different set of goals and faces different constraints when conducting a DBMS benchmark.

1) DBMS companies conduct benchmarks to demonstrate the strengths of their products (and persuade potential customers to buy). They will expend massive amounts of effort to tweak, tune and optimize a test to wring the last ounce of performance from a particular configuration. They will try to hide any product limitations. They like to set up and perform preliminary testing in private, bring the customer in to "witness" the test and then get the customer out quickly before anything can go wrong.

2) Computer equipment companies conduct DBMS benchmarks to show how well their machines perform. A DBMS benchmark is viewed as a necessary evil to accomplishing the real goal (sell the computer). They are not interested in accurately profiling all the strengths and weaknesses of several database management systems, but rather, they want to show the customer just enough to close the sale. Sometimes there are "strategic relationships" between computer equipment companies and certain DBMS

vendors, in this case the computer equipment company may try to direct the customer to select one particular DBMS product. In this case the computer company will begin to tweak and tune to make one product look

3) Magazines, Newspapers, and other publications will sometimes sponsor benchmarks to be included in product reviews. The products selected for measurement and the timetable for the reported results are all governed by the editorial calendar of the publication. If a customer is interested in some other product or needs to make a decision before the article is published he's out of luck. The reported figures are usually brief because the publication will not have space to devote to a lengthy report of highly detailed information. The technical people conducting the test cannot investigate every significant issue because they have a deadline that must be met.

4) Universities and other academic institutions will sometimes report results from database benchmark tests. Occasionally, these tests are often part of a research project linked to a particular hardware or software product. Sometimes the tests compare experimental or research versions of machines to DBMS products. These tests can be very detailed and specific in some areas and completely omit areas not relevant to the particular research topic. In any case the products tested, platforms used and timetable of the tests are controlled by the researchers.

5) Finally, customers will conduct their own DBMS benchmarks to try to profile the strengths and weaknesses of a database management system. This is like "do it yourself bridge design." At first glance the process seems simple but there are some subtle issues that can cause serious problems (and occasionally complete failure). It is significant to note that a customer will typically involve one or more database vendors and a computer hardware vendor in this process. These organizations will not encourage the customer to conduct more thorough and detailed tests, because such tests take longer and are more likely to uncover "problems" that might kill the sale. The customer will be encouraged to "hurry up" the testing process and make the selection.

Some important areas of database benchmarking are not being met by any of these existing methods.

The market needs an experienced database benchmarking organization that is not tied to any hardware or software vendor. This organization could run tests on computers selected by the

customer, with DBMS products chosen by the customer, according to time schedules specified by the customer. The independent organization's goal would not be to sell a computer or a database package but rather to learn and report the truth to the customer.

The organization should offer a benchmark that was written by a single group of programmers with essentially identical syntax for all popular DBMS packages. The benchmark should be run in its "standard" form before modifications or tuning is performed. Then the benchmark can be run again after "tuning" is completed by the computer or DBMS vendor.

By comparing the original times to the times after tuning it will be evident how much performance improvement resulted from the tuning. By comparing the original code to the modified code it will be clear how much custom programming effort is required to get the performance improvement.

The benchmark should measure many different aspects of database performance at many different "user levels" and database sizes. The benchmark must report the results in detail so that both the strengths and weaknesses of the DBMS are uncovered.

The benchmark should be run in its standard form for multiple database products on a single computer platform to show relative performance of the DBMS packages. The benchmark should be run in its standard form with the same DBMS package on multiple platforms to show relative performance of the computer equipment.

Neal Nelson & Associates has developed a database benchmark to address these needs.

2. Description of the Benchmark Operation and Methodology

Since the benchmark is coded in industry standard SQL, it can run with almost any database product that supports SQL on many platforms ranging from PC LANs to classical mainframes, including the wide variety of Unix-based machines that are popular in the marketplace.

The benchmark is run by using Remote Terminal Emulation benchmarking techniques. With this methodology two computers are connected "back to back" so that characters transmitted from ports on one machine (the Remote Terminal Emulator or RTE) are sent into corresponding ports of a second machine (the System Under Test or SUT).

Command files called "scripts" are executed on the RTE that transmits keystroke sequences that emulate users performing various database activities.

The RTE test methodology allows the test suite to be run with one user, two users, three users and so forth until a database/computer combination has been adequately stressed.

The RTE testing technique is the ideal method to test an application program such as a database. It recreates the exact loads caused by users running commands from terminals (including terminal input and output activity).

2.1. Running The Test

The first step is to create the database tables and their associated indexes. The database is then loaded to the first "user level". (Each user level requires approximately 30 megabytes of disk space.)

With the database at the single user level, one user performs the various database functions that make up the database benchmark. Besides timing each function, the RTE validates the result of the function to insure that it was processed correctly.

When the single user test has completed, the database is loaded to the two "user level" or approximately 60 megabytes. The number of active users performing each function is increased to two. When there is more than one active user, the execution of the various tasks is controlled so that each user runs his task at exactly the same time as all the other users.

The sequence of increasing the database size and adding an active user is repeated until the database reaches a particular size or the response times become excessively slow.

3. Description of the Database Design and Organization

The benchmark uses a test database that was patterned after a job-costing system, a common business application. The Benchmark's test database has 13 tables such as Customer Master, Employee Master, Department Master, a Transaction Table, and several cross- reference tables.

The benchmark suite includes a set of programs that generate fixed data for the various tables. These programs create known relationships between files (such as 100 jobs for each customer with 80 "open" jobs and 20 "closed" jobs). They load the data fields and key fields with values that allow automatic accuracy checks of the results returned by the database; for example, with a range of customer numbers that include ten customers, a select for "closed" jobs should return 200 rows whereas a select for "open" jobs should return 800 rows. The database benchmark suite automatically logs error messages when there is an incorrect result.

The test database incorporates compound keys, large rows (one at least a kilobyte), and large tables (ranging from 20,000 to two million rows, depending upon the number of emulated users). For example, the "Customer Master" table occupies approximately 540 kilobytes of disk space for a one-user test and 2.16 megabytes of disk space for a four-user test. The total storage requirement for the database is approximately 30 megabytes for each emulated user. The actual space required varies with each database product.

The known distribution of the database key fields allows different tests intentionally to create or prevent "hot spot" accesses. A "hot spot" occurs when many users access data elements that are located in a comparatively small portion of the database. Often, these areas are copied into the computer's disk cache buffers, which creates the false impression of superior database performance because no disk I/O is taking place.

The table below details some important aspects of the database tables.

Table Name	Number of Columns	Number of Keys	Key Types*	Key Length	Row Size
Transaction Table	108	3	N	11	440
			N,C2	21	
			A	12	
Job Master	19	2	N	10	720
			A,C2	22	
Customer Master	21	2	A	12	1,032
			A	40	
Sales Rep. Master	13	1	A	12	236
Employee Master	13	2	A	12	248
			A,C2	24	
Department Master	2	1	A	12	44
Operations Master	4	2	N	8	44
			N,C2	16	
Work Center Master	3	1	N	8	40
Qualification Master	2	1	N	8	36
Job Comments	8	1	A,C3	28	1,000
Customer/Sales Rep	4	2	A,C2	24	32
Cross Reference			A,C2	24	
Department/Work Center Cross Reference	2	1	A,C2	16	16
Employee/Qualification Cross Reference	8	1	A,C2	1	420

* N=Numeric, A=Alphanumeric, Cn=Compound Key n=number of key parts

4. Sample Queries

The database benchmark suite consists of 59 measured functions. These functions are divided into the following six major groups:

1) Keyed Queries: These queries are designed to test a database's ability to access rows and groups of rows through indexes.

2) Sequential Queries: These queries are designed to measure access to the database when the requested information is not identified by an index.

3) Bulk Functions: Bulk functions such as import, export, batch insert, batch delete, are included in the benchmark suite because they are frequently used and sometimes cause significant degradation.

4) Data Manipulation: Locking of tables and records is a vital concern in a multi-user environment. The benchmark suite includes a variety of functions such as insert, update and delete that require lock management services. The inherently multi-user design of the tests reveals the type and efficiency of the database's locking facility.

5) Transaction Processing: This group performs a series of operations grouped together to form what is commonly called a transaction.

6) Aggregate Queries: This group consists of SQL statements that contain one or more aggregate functions such as "sum", "avg", or "count".

The tests are designed to answer many questions, such as "How dependent is the database product on specific programming techniques?" or "How will system performance degrade as the database becomes larger?". To answer the first question, queries are coded in different manners. For example, the two queries illustrated below are constructed using two formats where one query incorporates a nested format and the other a non-nested format.

Nested version

```
select   distinct emepno, dmdpno, omqlcd, qmdesc
from    empmst, dptmst, qlfmst, oprmst
where  omqlcd = qmqlcd and
          dmdpno = emdpno and
          emepno = '000100000000' and
          qmqlcd in ( select  eqqlcd
                           from   empqlf
                           where eqepno = '000100000000');
```

Non-nested version

```
select   distinct emepno, dmdpno, omqlcd, qmdesc
from    empmst, dptmst, qlfmst, empqlf, oprmst
where  omqlcd = qmqlcd and
          qmqlcd = eqqlcd and
          dmdpno = emdpno and
          emepno = eqepno and
          eqepno = '000100000000';
```

Although the end result is the same, the processing time may vary significantly. In one test case with four active users, the nested version averaged 12% slower than the non-nested version. This test is interesting because the database documentation has never explained that one technique is superior to the other.

Another aspect of the database benchmark is to gradually increase the complexity of a query. For example, a baseline measurement is taken where a query specifies one table. Five more tables are then added and another measurement taken. Another five tables are added to the query (for a total of 11) and another measurement taken. Using the 11 table join query, another measurement is taken, this time reversing the order in which the tables are specified. Finally, the 11 table join is executed through a view to show the relative speed when accessing data through a view as compared to explicit SQL statements.

Even though queries make up an important part of any database benchmark test, common database functions such as import and export are also measured. Other tests are included that measure the overhead of the sort function, and the impact of using secondary and compound keys.

Backup is also an important, but seldom-discussed, aspect of database management systems. Since most popular DBMS's encourage the use of "raw" disk devices and archiving of data to some external medium, the benchmark suite tests the speed and efficiency of the backup utilities.

5. Sample Test Results

This section contains two sets of tests that illustrate the type of information collected by the Neal Nelson Database Benchmark. We have intentionally omitted specific information such as brand, model, capacity, software version, access time, operating system release and so forth. It is not our goal to draw any conclusion about whether a particular product is better or worse, but rather to demonstrate that significant differences exist that can be measured and reported with the database benchmark.

The benchmark is designed to be configurable by customers at the time of a particular test. In particular the typing speed and think delay can be set to reflect a customer's environment. The samples shown here were collected with think time set to zero to create maximum stress on the system being tested. In these tests, response times of several hundred seconds are resulting from only four active users.

The database and queries are designed to allow 1,000 emulated users. The number of actual users that a particular customer may choose to emulate will depend upon the size and speed of the system being tested and choices for think delays and typing speed.

The first set of sample test results shows a comparison between two different database management system software products on the same computer. The sample shows ten different types of activity for one through four simultaneous users.

The results for DBMS 1 are plotted with solid lines and the results for DBMS 2 are plotted with dashed lines.

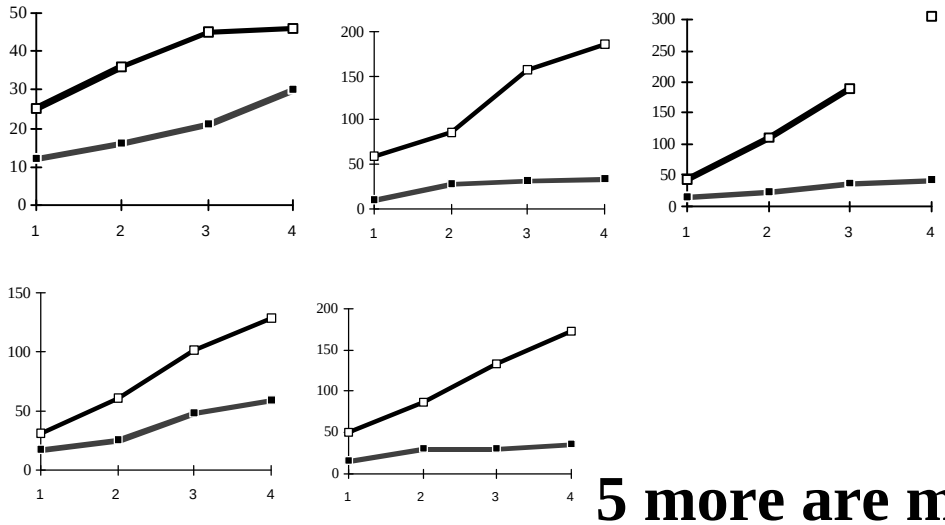
Both of these databases are major products. They are definitely in the top three or four in the market. They would be viewed by many people as being "comparable" and yet in seven of the ten tests shown DBMS 1 is significantly (53 to 931 percent) faster than DBMS 2.

The Sequential Query (Non-nested) versus Sequential Query (Nested) is interesting since identical functions are being performed and yet by changing the syntax of the query, DBMS 1 went from being 4 to 8 times faster to being up to 2 times slower.

Table 1: Comparison of two SQL Systems

System A  **System B** 

(headings and legends missing)



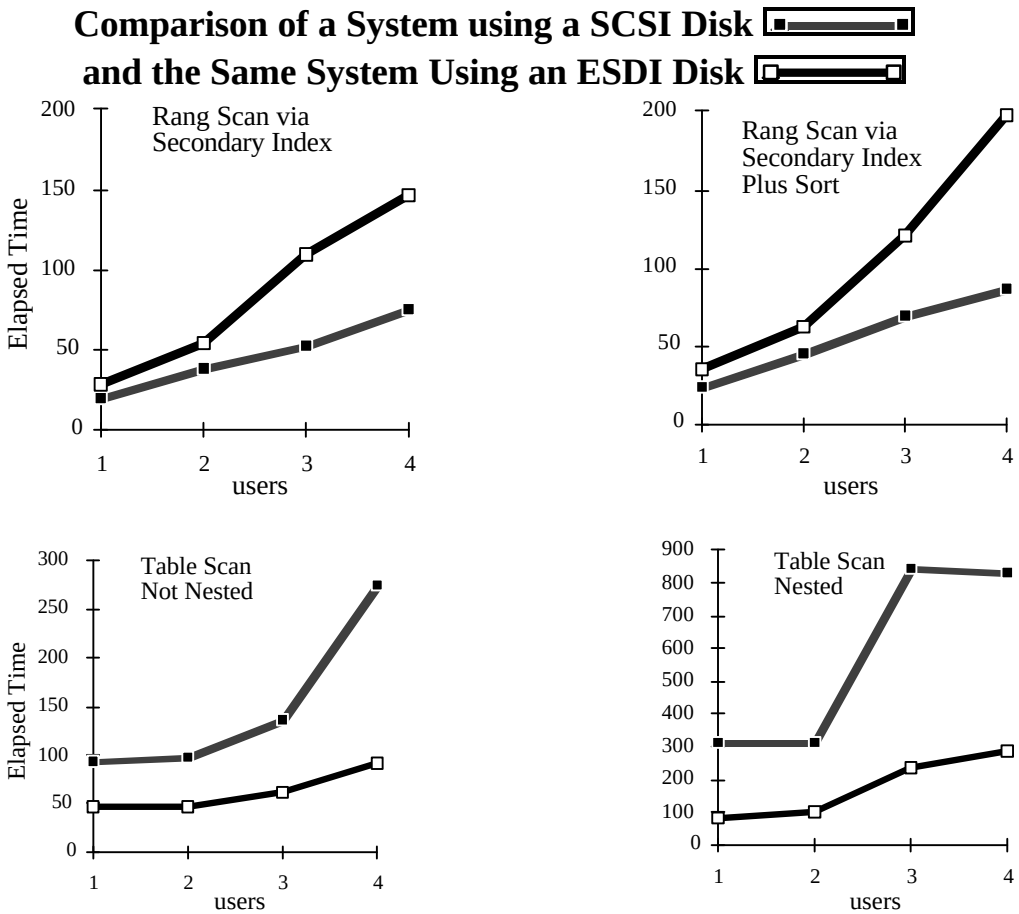
5 more are missing

The second set of sample test results shows a comparison with the same DBMS product running on the same computer with two different disk drives installed for the two test runs.

When this test was conducted the same ten measurements were taken that were listed in the previous tables. The first six sets of measurements showed almost no difference between the ESDI and SCSI drives. These six graphs have been omitted to save space.

It is interesting to note that for the next two functions performed (keyed queries) the ESDI disk appears to be significantly faster and yet on the last two queries (which are both sequential scans of the entire database) the SCSI drive is two to three times faster than the ESDI drive.

It must be remembered that there are many factors that affect the throughput of a disk subsystem (controller, driver, etc.) and the following data is presented only to illustrate that differences exist that can be measured and reported by a database benchmark.



6. Observations and Findings from Customer Uses of the Benchmark

The following examples show how the Database Benchmark was used to solve specific business problems. The tests not only clocked how quickly each package executed a given problem; they uncovered a number of "quirks" that could affect a user's evaluation of a given package.

6.1. Army Database

In June 1989, the U.S. Army used the Neal Nelson Database Benchmark™ to compare four database products on two different computers. The following describes some of the findings:

- In one test, the benchmark driver emulates multiple users concurrently inserting rows into an indexed file. Running this test showed that one popular database system locks the entire table when processing an insert, and keeps the lock until the transaction commits. In some instances, this “quirk” raised the processing time for a single insert to as much as 25 seconds.

- The Benchmark creates a table that, on most machines, exceeds the storage space of a single disk. The particular test was executed in four stages, with the size of the database doubled at each stage. By the third stage of the test, however, we discovered that two of the four products under consideration would not allow tables to span physical devices, and so we could not run the remainder of the test for those two products.

- SQL allows the user to structure queries in different formats to arrive at the same result. To address this issue, the Benchmark compares nested queries with non-nested queries. This test provided unexpected results. When processing a nested query, some products actually reverted to searching the data sequentially, even when the key to the table is provided. In another instance, a product processed a query with sub-selects faster when keyed access was specified, but slower if the query searched the data sequentially.

- The Benchmark also compares the speed and efficiency of the database product's backup facility. One product did not support a backup of raw device files[1]. In addition, although the size of the database did not vary, the number of tapes required to backup 400 megabytes varied from 3 to 17 for different DBMS products.

- The Benchmark was apparently too rigorous for one database product. This product could not run the multi-user insert portion without core dumping, a discovery that initiated the first of what would ultimately be three product updates over the course of the test.

- At first, the test database included a 10,000-byte row; however, this test quickly uncovered the fact that one database product could not support a row larger than 2,048 bytes. To circumvent this problem, the row size was changed to 2,000 bytes.

- Because the test database includes tables with unique secondary indexes, it was discovered that one of the products would not support this feature.

- The test was planned with raw devices for all databases, because there is a consensus that this would deliver the best performance. One of the products did not support raw

[1]A "raw device" is sometimes preferred by DBMS vendors. With a raw device I/O is sent straight from the DBMS program to the disk bypassing any overhead provided by the operating system

devices so it was run with Unix file systems. The product using Unix file systems ran twice as fast as some other products using raw file systems.

- No product tested included a utility to check the integrity of the indexes.

Overall, it was found that the fastest product ran approximately two times faster than slowest product. We also found, however, that the fastest product executed almost as quickly on a lower-priced machine as it did on a machine that was almost five times more expensive.

6.2. Retesting of an Existing Application

The Neal Nelson Database Benchmark was used in January 1990 to test a new release of a database product that was being recommended to a large military organization. This organization had already invested thousands of man-hours to test this product, yet the Benchmark uncovered several problem areas with the new product, including the following:

- The multi-user insert test revealed that during heavy use, the DBMS updated the wrong pointer in the index table, which then corrupted the database. This bug had been in existence for three years (from the time the product was introduced).

- When two or more users selected the SQL module within five seconds of one another, the application would hang.

- A select statement with no "order by" clause returned a different number of rows than the same select statement with an "order by".

- The SQL program required a temporary space equal in size to the database in order to perform more complex queries.

- When the same queries were run with different formats (nested versus non-nested, etc.) sometimes the execution time would double.

- The multi-user-insert test caused memory violation messages.

As these experiences show, an automated tool with a standard set of tests is a very sensible approach to validate new software and updates of existing software. For example, on the day of a software briefing, a representative of the database vendor appeared with a new version of their product. The person at the military organization responsible for validating this product tested the latest version that morning and by the afternoon presented the new findings to the briefing panel.

7. Checklist to Help Determine the Validity of Database Benchmark Results

There are several major reasons why database benchmark results can be deceptive or misleading. The following list of questions will help customers determine if a particular set of database benchmark results provides significant information about the speed of the hardware or software product.

Each topic is organized into a major headline, explanatory text and a question which might be asked of a vendor.

Database Benchmark Source Code

The most fundamental concept of benchmarking (including database benchmarking) is comparison. The questions that are most frequently asked are "Which database is the fastest?" and "Which machine is the best choice for this database?"

A useful comparison cannot be made if different programs are used on different machines since any reported variance might be caused by the programming technique rather than by the machine or software product.

In some cases the computer or database vendor is allowed to write and/or extensively modify the benchmark program before a benchmark test. When this is done it is difficult to sort out the effects of programming style from effects caused by the machine or software product.

Question - Was the benchmark coded using essentially identical syntax, by the same individuals, for all appropriate hardware and software products?

Constant Test Platforms

The usefulness of a database benchmark is reduced if each different database product is tested on a different computer system. Reported variances between the results could be caused by either the software packages or the computer systems.

Question - Are database benchmark tests for a variety of software packages run on a single computer system?

Cache - Cache Size and Test Database Size

Many computers will keep some of the recently accessed disk sectors in main memory in a series of buffers that constitute a "cache". When an application program (such as a database package) requests data from the disk, the operating system first checks the buffer cache to see if the data is in memory. If so, the "read" is satisfied from cache and the physical disk read is avoided. Cached disk I/O is dramatically faster than actual physical disk I/O (one or more orders of magnitude). If a test database is so small that it will fit substantially (or entirely) in disk buffer cache, the benchmark results will indicate that the configuration is very fast.

However, when the database size is large enough that it will no longer fit in buffer cache (which is normally the case), the actual performance will be quite slow compared to indications provided by the benchmark.

Question - Is the size of the test database and/or the configuration of the test computer such that most (or all) of the disk I/O will take place in and out of cache?

Functions Measured and Reported

Database products must perform a large number of different tasks such as:

1) Import Batches
2) Export Batches
3) Interactive Inserts
4) Interactive Deletes
5) Lock, Read, Rewrite, Unlock
6) Change Key Fields (Update Indexes)
7) Commit Transactions
8) Random Keyed Queries
9) Queries by Range
10) Sequential Queries
11) Sorts
12) Aggregate Functions Such as Count and Sum
13) Backup

Database benchmarks that perform only a few of these activities and/or report only a few numbers will not provide an adequate report of the product's performance in different

categories. A database product might have been "tuned" to provide excellent performance in random reads and inserts but deliver terrible performance in sequential queries and sorts.

If a particular benchmark performs only a few functions or if a variety of different functions are used in combination and only a single result reported, it might be impossible to identify that some particular activity is quite slow.

Question - Does the database benchmark measure, and report separately, a wide variety of different database activities?

Query Optimization

Most database management systems have built in query optimizers that are supposed to insure that a query is performed in the fastest and most efficient manner. SQL allows a wide variety of "legal" ways to state a query, but when the query is actually performed there can be significant differences in speed depending upon the sequence of selects, sorts and other functions.

The Query Optimizer is supposed to automatically identify the best way to perform a given query and automatically reformat the request into this most efficient structure.

If the Query Optimizer is inadequate, different users will experience dramatically different levels of performance, based on the syntax of their SQL queries. Users and programmers will be forced to learn what syntax delivers superior performance from a database package, and manually code their work to obtain efficient execution.

Question - Does the database benchmark contain identical tasks coded with different SQL syntax to help determine how well the query optimizer works?

Increasing Test Database Size, Increasing Numbers of Users

Sometimes database products perform quite well up to a certain number of users, or up to a certain database size, and then their performance degrades dramatically.

If a database benchmark reports it results at only one database size, or with only one set of emulated users, it is impossible to know if the reported figures were before or after a possible fall off in performance.

Question - Does the database benchmark report performance figures for a range of database sizes and/or a range of simulated users?

Emulate a Real World Application

A database management system is the foundation of a wide variety of general business activities such as order processing, accounts payable and payroll. A database benchmark should emulate some common business activity to provide the most meaningful test results.

Question - Does the database benchmark perform functions similar to the activities required for daily operation of your business?

Sufficiently Complex

If a database management system is going to break down, it will most likely happen during a time of high load or high stress. A benchmark should provide enough activity to cause significant stress during the test, and possibly uncover problem areas before the product is installed.

Question - Does the database benchmark generate enough stress to uncover problem areas?

8. Conclusion

Vendors, scholars and journalists have all focused on different parts of the performance puzzle and have developed different methodologies to try to "solve the database benchmarking problem".

Neal Nelson & Associates has developed a database benchmark that supplements and extends other benchmarks by measuring performance factors that have not been previously reported.

The Neal Nelson Database Benchmark™:

1) Was written with standard SQL commands by a single group of programmers

2) Can be run on a single platform with multiple DBMS packages to highlight software differences

3) Can be run on multiple platforms with the same DBMS package to highlight hardware differences

4) Can be run before and after "tuning" to show how much performance improvement results from what level of changes

5) Can be run for the machines and software packages specified by the customer in a time frame defined by the customer

6) Is sufficiently complex and scalable to stress virtually any hardware/software combination

7) Provides reports of degradation as the "user load" increases up to a configuration's breaking point

8) Provides detail test results for scores of different functions to show both the strengths and weaknesses of a DBMS product

Further information about the Benchmark can be obtained by contacting:

Neal Nelson & Associates
35 East Wacker Drive
Chicago, Illinois 60601
(312) 332 - 3242