Informix Goes GUI SEE PAGE 34

DBMS

CLIENT/SERVER COMPUTING

NOVEMBER 1993 VOLUME 6 NUMBER 12

McGoveran Says "No" To Useless Benchmarks

Pushing for Performance

Roti Reports On XDB-Server

Networking FoxPro
PAGE 93

$2.95 (Canada $3.50)
11
0 38351 75230 9

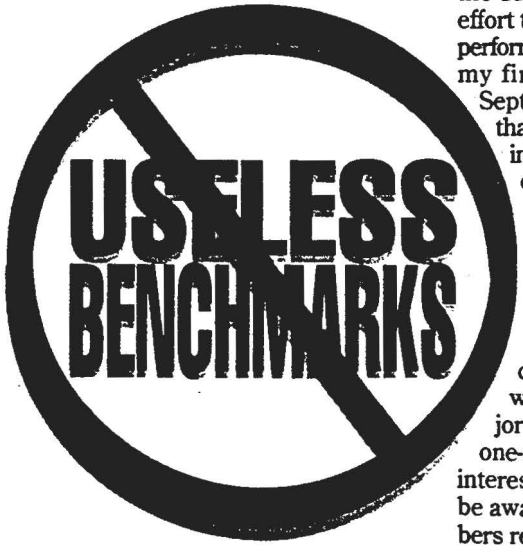
REVIEWS
SuperNova, Informix Tools

INTERVIEW
A Quantum Leap With Dick Pick

USELESS BENCHMARKS: Just Say No!

EVERYONE WHO IS INTERESTED IN DBMS PERFORMANCE SHOULD BE AWARE OF WHAT TPC BENCHMARK NUMBERS REALLY (DO NOT) MEAN.

BY DAVID MCGOVERAN



Having spent a significant portion of the last ten years consulting on relational DBMS performance modeling and tuning, unaudited benchmarks, postmortem audits, and complex relational application design and development, I assumed that the TPC benchmarks were a reasonable effort to provide users with useful price/performance information. After I completed my first audited TPC benchmark in September 1991, I became convinced that I was wrong. That benchmark was intended to meet the Transaction Processing Performance Council (TPC) Benchmark B specifications.

I thought at first that I should share what I learned with my clients. The actual experience was so illuminating that I immediately wrote the first version of this article instead. Interestingly, the paper was rejected at that time by every major industry publication as being "too one-sided." I disagree. Everyone who is interested in DBMS performance should be aware of what TPC benchmark numbers really (do not) mean.

First, the reader needs to understand a bit about the TPC Benchmark B specification, and the TPC, its history, and current status. The TPC benchmarks are based on certain early benchmark efforts, notably ET1, TP1, and DebitCredit. Tom Sawyer (then of Codd & Date Consulting, and now Performance Metrics) and Omri Serlin of ITOM International first attempted to standardize DebitCredit with a standards document. The TPC was subsequently founded by Omri Serlin in 1988. The council consists primarily of system and database vendors. (Virtually all the major U.S. vendors are represented.) The TPC issued the first standard, TPC Benchmark A, in late 1989. TPC Benchmark B was published in mid-1990. TPC Benchmark C was issued in 1992, and TPC

David McGoveran has been President of Alternative Technologies, a Boulder Creek, California database consulting firm, since 1976. He is author (with Chris Date) of A Guide to Sybase and SQL Server, *and publisher of in-depth technical studies of major DBMSs, The Database Product Evaluation Report Series.*

Benchmarks D, E, and Client/Server (C/S) are expected to be released in 1994. The critical comments in this paper apply to Benchmarks A, B, C, and most probably will apply to D, E, and C/S as well.

Both the TPC-A and the TPC-B benchmarks use the same simple database (see Listing 1) and simple transaction profile. TPC-A includes terminal emulation, whereas TPC-B permits the use of batch transaction generation. Typically, benchmark tests are run with the system in isolation; only four tables will exist in the database, only one type of transaction (consisting of three record updates, one record insert, and one record retrieval — as shown in Listing 2) will be submitted during the test, and no extraneous processes will run on the system during the test.

A benchmark is run by a benchmark team and has a sponsor. (These are typically part of the same company, usually the DBMS vendor.) It applies to the hardware and software system in a particular configuration, called the "system under test." Typically, a benchmark effort will consist of numerous "runs," with the better runs being used as an indication of better tuning. During an audit, several runs may be made and, assuming the auditor has no objection, only the best run will be reported. After the benchmark has been audited by an independent auditor, the results are written up in a report (called a Full Disclosure report). The intent of this report is to allow others to replicate the results. This document is submitted to the TPC for a 60-day compliance review, whereupon it is either rejected or accepted. In general, discussions held during TPC meetings are secret; the rules of membership require that they be kept in confidence. Supposedly, this is to allow vendors to engage in "free discussion without fear of embarrassment."

While one or two of the remarks in this article are specific to TPC Benchmark B, they generally apply to past, present, and future TPC benchmarks as well, and are directed at the TPC. I believe that vendors will not be accused of "cheating," running "benchmark specials," and the like if the TPC does its job. This article is intended to pressure the TPC and its vendor members into creating a benchmark committee we all can trust; one that does not need secrecy.

■ During an audit, several runs may be made, and, assuming the auditor has no objection, only the best run will be reported.

Roughly, my criticisms of the TPC and of TPC benchmarks fall into four categories, as follows:

1. Benchmark Design Issues
2. Interpretation of Results
3. Compliance Issues
4. Criticisms of the TPC

I discuss specific criticisms in the following sections.

Benchmark Design Issues

Statistics. Although there is good reason to be understanding when criticizing the TPC, I find the TPC's failure to specify good experimental measurement practices unforgivable. During a TPC benchmark, the benchmark team collects and reports counts of transactions performed by the system under test once it has settled down into a "steady state" period and, among other things, computes a measure of concurrency. Any freshman college student who takes a course in laboratory science (for example, chemistry) learns how to characterize or estimate errors, how to calculate standard deviations and specify confidence factors, and how to describe measurements so that they are repeatable. It seems that, with respect to reporting benchmark results, TPC members forgot all this (or never learned it).

As a result, there is no specification of the number of data points that must be taken (only the minimum number of intervals is given), no characterization of the accuracy with which intervals are measured (yes, accuracy is relevant when making discrete measurements), no calibration procedure for the measurement subsystem, no measure of confidence in the results, no measure of variance, and so on. Were these characterizations to be included, they would provide some assurance that the benchmark results would be repeatable.

Of far greater concern is the lack of any definition of what is meant by "steady state." For example, this could be given as the requirement that the system give the reported TPS (transaction per second) value in the mean, with a variance of no more than two standard deviations. At the very least, and under the assumption that systems cannot be required to reach any predefined steady state, the benchmark performer should provide a characterization of the variance during steady state measurements. Nothing of this sort is done.

Similarly, it is possible to measure the time-dependence of variations in reported numbers, and to decompose that time-dependence mathematically so as to determine if it is periodic or essentially random. These characterizations can help a user evaluate the stability of system performance and help benchmark teams determine if there are important variables in the configuration that remain uncontrolled. The results of this method also serve as an indication of how predictable

LISTING 1. The TPC-B database specification

```
                    HISTORY
Account_ID, Teller_ID, Branch_ID, Delta,
Timestamp, Filler_to_50_bytes

                    ACCOUNT
Account_ID, Branch_ID, Account_Balance,
Filler_to_100_bytes

                    TELLER
Teller_ID, Branch_ID, Teller_Balance,
Filler_to_100_bytes

                    BRANCH
Branch_ID, Branch_Balance, Filler_to_100_bytes
```

LISTING 2. TPC-B Transaction Profile

```
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 Aid, Tid, Bid, Delta, Time_stamp to History;
    Update Teller where Teller_ID = Tid;
        Set Teller_Balance = Teller_Balance + Delta
        Write Teller_Balance to Teller
    Update Branch where Branch_ID = Tid;
        Set Branch_Balance = Branch_Balance + Delta
        Write Branch_Balance to Branch
COMMIT TRANSACTION
Return Account_Balance to driver
```

Note that statements can be in any language and in any order.

performance is likely to be, especially when a user's expected system configuration differs from the benchmark configuration. Again, the TPC reports do not contain such information.

Unrealistic Configurations. In developing production applications, the resources of the entire environment are normally considered. Thus, an IBM system may use CICS, and a DEC system may use ACMS. If these are deemed too expensive, an organization may use custom software to achieve the same effect. The current interpretation of the TPC-B disallows such efforts. On the other hand, the particular configuration of the DBMS chosen for benchmark purposes may effectively disallow backups or forward recovery procedures as they would be required during normal operating conditions. Users need to be informed of any disallowed features that might have improved benchmark results, and, likewise, of the key potentially negative "side effects" the particular configuration and tuning might have on a particular application.

High Cost. Many factors contribute to the high cost of a TPC benchmark. Among these are the costs of software, hardware, the driver, lengthy debugging times, system tuning times, cost of technical personnel, auditor costs, and report production costs. Given these, it is not unusual to measure costs in the hundreds of thousands of dollars. One would hope that these costs would be much less if a user were simply replicating some previously run benchmark. Unfortunately, this is not the case because of the time required to rediscover necessary benchmarking information. (See the section titled "Inadequate Reporting" later in this article.)

Furthermore, most of these costs are not reported. Thus, somewhat better benchmark results might be obtained only at a much higher relative cost for tuning and driver development. How can users make comparisons if such costs (both time and expense) are undocumented?

In some cases, benchmark costs are arguably artificially high. For example, the specification requires actually having on line, in the tested configuration, sufficient mass storage to operate for eight continuous hours even though the actual measurement interval need be only 15 minutes (clause 7.1). There is no obvious reason the amount of storage could not simply be computed rather than configured. Such costs do not make the benchmark accessible and it is not clear that any potential value of having these requirements exceeds the costs they impose.

Tuning Complexity. The average user simply does not have enough information to tune the system the way it is done by the typical benchmark team. Very often, privileged knowledge of the internals of the DBMS, operating system, or hardware are required. These facts are known to the product designers, and occasionally, a few independent consultants. It is extremely unlikely that the reported results could be obtained from readily available information about the DBMS and operating system alone. The average user does not have the requisite experience in performance tuning; even after ten years of consulting, I find each such effort to be an intense learning experience. All information used in a benchmark should be available publicly so that users can benefit from such knowledge. Essentially, a "lab notebook" that describes the tuning process is needed. It is surprising that vendors do not see the advantage of offering such information to their customers, especially when they seem so eager to offer (perhaps unintentionally) misleading benchmark results.

■ The average user usually does not have the privilege of working with such severely isolated systems.

In addition, these benchmarks are generally run with the system isolated from nonbenchmark activity. The average user usually does not have the privilege of working with such severely isolated systems. The typical application transaction mix, in conjunction with the significantly more complex databases that are used in practice as compared to benchmark databases, are sufficient to make a benchmark run in isolation all but worthless to users. Users need some method to measure how lack of system isolation will affect benchmark results; the TPC should not assume isolation is possible. The exact system configuration used in a benchmark is highly unlikely to be used in practice. It thus becomes extremely important that users are helped to assess the robustness of a product and the meaningfulness of benchmark results.

Interpretation of Results

Lack of Comparability. The numbers produced by different runs of a benchmark are not comparable, let alone with those produced by different benchmarks. There are two reasons for this: First, the numbers reported in a Full Disclosure report are not properly characterized as empirical measurements, and second, the specification and configuration are very loosely defined. They simply do not measure the same thing quantitatively and probably do not even measure the same thing qualitatively. A benchmark using a database is fundamentally different from a hardware benchmark; it is inherently more complex. Suggesting the contrary by comparing benchmark numbers, even implicitly, is naive. I certainly would not go so far as to say that two sets of TPC numbers can even indicate relative performance, even if they are qualified by a trademarked phrase like "Benchmark B TPS."

No Measure of Product Capabilities. Unless all of a product's features can be used in testing performance, it does not make sense to treat the TPC numbers as a measure of a product. The features and techniques that account for performance differ greatly between products, a fact of which everyone who has tuned the performance of multiple DBMS products is aware. The factors that must be taken into account for tuning are often subtle.

As noted in the next section (see "Compliance Issues"), the product features, application techniques, or system configurations that would make a significant difference in performance or concurrency levels can be disallowed arbitrarily by the auditor or the TPC reviewers in a TPC benchmark. For example, both database triggers and the use of row identifiers have been disallowed without (in my opinion) reasonable justification and certainly without any publicly available written explanation by the TPC.

In order to measure a well-defined characteristic of a system, it can be desirable to force a particular method of implementation. In such circumstances, one attempts to define a test that eliminates unrelated and unknown parameters (noise), or reduces their impact on the measurement. Unfortunately, the TPC specifications are so loose that none of the measured characteristics can be said to be well-defined.

Inadequate Reporting. For reported results to be reproducible, users need to know what was done during the benchmark and how. They need to know which portions of the configuration are sensitive and which are robust. The TPC Full Disclosure reports typically do not record such information, nor can users infer it from the information that is recorded. Even worse, reported database code (for example, SQL) is sometimes syntactically incorrect and program code (such as the code used in the database loader program or the driver) will not even compile. When asked for this information, some vendors (benchmark sponsors) have refused, calling it "proprietary."

Any independent party should be able to repeat a published benchmark and comprehend why particular tuning decisions were made. The preferable solution is for the TPC to offer the required pub-

lications. Alternatively, sponsors should document and offer the necessary information. Neither solution is available today. Indeed, it is doubtful that the benchmark teams know and could reproduce the results in a second independent benchmark if some time were allowed to elapse — even given access to their own notes!

The report provided by the TPC — which is devoid of (1) the reasons for implementing the benchmark in a particular manner and (2) an analysis of functional trade-offs among the possible implementations — is not sufficient. A comprehensive analysis of the results, listing the trade-offs for each tuning, pricing, and configuration decision is needed. Benchmark reports should contribute to the body of available knowledge about how a product works and how best to use it. If they do not, they are de facto imprecise and unscientific. A complete report need not greatly increase the cost of running a benchmark. Much of the needed information should be relatively constant and need only be produced once. The remainder could be produced semiautomatically, requiring a modicum of manual attention.

Compliance Issues

Auditor Comprehension. The auditor plays an important role in the TPC, much like a financial auditor does with respect to a corporation. Unlike a financial auditor, TPC auditors must establish their own procedures for verifying compliance. This situation arises in part because of the lack of specificity of the benchmark, and in part because of the wide variety of systems the auditor is likely to audit. But it is certain that no auditor is likely to comprehend all the myriad intricacies of every operating system, DBMS, programming language, and custom driver design combination encountered.

Given so many variables, how can the auditor be expected to develop a compliance test suite that is unbiased and without serious loopholes? While I have great respect for some TPC auditors, I would not expect such superhuman feats from anyone. Furthermore, without uniform guidelines, why should one auditor's compliance test suite be as stringent as another's? The TPC should clearly specify audit procedures and approve and document exceptions to these procedures where they are judged necessary. In principle, it should be possible to fully automate a good audit procedure.

Auditor Discretion. Because of the lack of a tight specification and the lack of a uniform set of compliance tests, the auditor often must to make judgment calls. In fact, the auditor can simply allow or disallow the use of a product feature, a test configuration, or a particular test run, virtually on a whim. Unless the test sponsor is willing to engage a different auditor (perhaps after having already expended considerable resources with the current auditor), the decision — in practice — is final. It is unconscionable that benchmark auditors and sponsors are placed in this position. This will-o'-the-wisp character of audited TPC benchmarks is alone sufficient to make them useless. Certainly, the need for ad hoc and undocumented judgement calls by a human auditor should be minimized. Auditing may become a little better in the near future; TPC auditors could become licensed and paid by the TPC (rather than by test sponsors) in October 1993 if a pending proposal passes. (Given the record of the TPC on previous proposals to make improvements, I won't hold my breath.)

Serializability. Transaction isolation requirements are important for ensuring that benchmark results do not take advantage of either the particular transaction profile, or of the fact that the test is run with the system isolated. Clause 2.4.1 of the TPC-B specification states these as follows:

"Operations of concurrent transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed in some order.

This property is commonly called serializability. Sufficient conditions must be enabled at either the system or application level to ensure serializability of transactions under any mix of arbitrary transactions, not just TPC Benchmark B transactions. The system or application must have full serializability enabled, i.e., repeated reads of the same records within any committed transactions must have returned identical data when run concurrently with any mix of arbitrary transactions."

The specification then goes on to provide isolation tests, given that the DBMS uses conventional locking schemes (clause 2.4.2). These tests are flawed in two respects. First, any number of TPC Benchmark B (or A) transactions are intrinsically serializable. It is only when "any mix of arbitrary transactions" is run concurrently with the TPC Benchmark B transaction that the question of serializability has any meaning. Second, the TPC Benchmark B specification gives no specific requirements for systems that implement nonstandard locking, and isolation schemes. It simply requires the test sponsor to disclose those locking and isolation techniques and the test used to confirm serializability.

Under a generous interpretation, this would require that nonstandard locking schemes be fully disclosed by the sponsor (even those that are "proprietary"). In addition, the TPC would have to place significant emphasis on acceptance or rejection of the tests invented by the sponsor to confirm serializability, and would have to publicly report its justification of acceptance. Only in this way could users and technical analysts meaningfully interpret the benchmark results. Unfortunately, neither of these conditions has been met to date.

To my way of thinking, any benchmark that is run with the DBMS enforcing anything less than full serializability is in violation of these clauses under any reasonable reading. Many published benchmarks are run without ensuring serializability; it is easy to define a "mix of arbitrary transactions" that will violate serializability given the configuration tested. These benchmarks should be withdrawn voluntarily or the TPC Compliance Review Committee should throw them out.

As one subterfuge, the benchmark team or auditor may interpret the phrase "sufficient conditions must be enabled at either the system or application level" to mean that applications involving non-TPC transactions could be written so as to ensure they are isolated from the TPC transactions. But surely this is a violation of the intent of the clause; this phrase clearly refers to the benchmark system and benchmark application level and not to some other application level. Even more important, numbers reported from these benchmarks completely ignore the impact that such applications would have on concurrency.

Enforcing this clause could have a serious negative impact on reported TPC numbers. If serializability is enforced at the system level, some row-level concurrency schemes will be disabled, so that concurrent updates are prevented at the table level. In loosely coupled cluster configurations, system-level serializability enforcement can also have a detrimental impact on cache management schemes. The only other alternative requires programmer and ad hoc user intervention. Each transaction in the environment must enforce stringent isolation by using, for example, a combination of explicit table locking and special SELECTs that lock accessed data for the duration of the transaction (such as, SELECT FOR UPDATE). Obviously, table locking prevents concurrent updates at the table level. The required special SELECT statements often cannot be used with a wide variety of important SELECT constructs, such as DISTINCT, GROUP BY, UNION, INTERSECT, COUNT, MAX, MIN, SUM, or AVG. These restrictions effectively disable concurrency for all but very special mixes of transactions.

Suppose that, for a DBMS with an affected row-level concurrency scheme, a transaction is needed that conditionally updates a single row in one of the TPC-B tables based on the sum of a column's values from a small subset of the rows in that

table. Ensuring serializability of such a transaction mix would require locking the entire table for update (because SELECT FOR UPDATE could not be used). While that transaction ran, no concurrent TPC-B transactions could complete! This is simply a fact of life for all DBMSs. Unless there is a weakness in the concurrency control scheme, system-level serializability enforcement should not impact benchmark results except for the additional overhead associated with enforcing serializability. It would, however, put all DBMSs on a par with respect to the cost of maintaining database integrity.

Even if you accept the proposition that SELECT FOR UPDATE is a permissible method of ensuring serializability, this technique places a great burden on the individual user. To assume that no user would make an error and forget to qualify each SELECT with a FOR UPDATE clause ignores the purpose of system-enforced serializability. It throws away more than a decade of research into, and implementation of, automatic concurrency-control technology.

Similarly, I cannot accept the point of view that regards the specification of tests of serializability as too difficult. A concurrency-control scheme can be proved to be either capable or incapable of serializability enforcement. In fact, it is the only isolation level that can be defined objectively (that is, so that possible anomalies are not system-dependent) and which, therefore, can be used as a condition for comparison of concurrency levels. It should be incumbent upon the vendor of the benchmarked system to provide such proof, and to show that the conditions for its validity are satisfied by the benchmark configuration.

Likewise, I do not find it reasonable that the TPC publishes a specification that implies TPC benchmarks enforce serializability while at the same time "interpreting" the clause into nonexistence behind closed doors. The TPC continually chooses a course of action that maintains appearances when faced with an opportunity to clarify and inform. This was done for TPC-A, TPC-B, and TPC-C, but the TPC chooses not to document its interpretation. Users cannot be expected to know that the publication contains wording that means one thing to TPC members and something else to everyone else in the world, or to judge the impact of such informal and unwritten interpretations. At the very least, the specification should remove any use of the term "serializability" because it uses the term improperly.

Criticisms of the TPC

Politics. The TPC has, on occasion, reinterpreted a benchmark specification after a benchmark was run. Faced with a Full Disclosure report, the members of the council have decided to disallow the results, not on the basis of some specifically violated rule, but based on subsequent interpretation of the intent of a rule. This would not be so bad if it were a serious attempt to produce a tight specification; then the specification could be modified and a new version produced with all subsequent benchmarks being required to comply with it. Considering the fact that the TPC consists of DBMS vendors, their efforts are more likely to be directed at improving their market position.

For example, a vendor's competitors will lobby (I use this particular political term advisedly) other council members to vote against the benchmark certification. When a TPC benchmark specification is reinterpreted or "clarified" after the fact, a new version of the benchmark specification is released. Such a new version is likely to eliminate some particular vendor's results. In fact, I have come to the conclusion that much of the wording in the benchmark specifications is written from the beginning with such intent, and that certain omissions exist as loopholes to serve some of the more powerful vendors.

Regardless of the degree to which such political maneuvering does or does not occur, a review committee made up of a vendor's competitors is clearly not in the best interest of the public. At the very least, the omissions in the specification discussed in this article are hard to justify as being in the best interest of the DBMS purchaser. Essentially, the TPC is nothing more than a tool of vendor marketing. It gives DBMS vendors who wish to engage in exaggerations and unethical competitive marketing tactics a pseudoscientific facade to hide behind. The TPC measurements of "transactions per second" amount to little more than marketing numerology. One sometimes wonders whether the numbers that a vendor publishes came first from marketing or from a benchmark effort.

Conclusions and Recommendations

The TPC needs to spend some additional effort in addressing the problems discussed above. There is no reason that benchmark design should not follow good statistical practices, that a benchmark should not be repeatable and uniformly interpreted, or that benchmark results should not be meaningful and useful to the public. The TPC should make every effort to fully disclose both the details and an analysis of a benchmark.

Users deserve to know the reasoning behind acceptance of a benchmark; how it comes to pass compliance, any criticisms leveled against the benchmark, and how they were answered. After all, it is the user who will spend company funds, who will put company data at risk, and who will gamble that TPC numbers are meaningful measures of performance when selecting and purchasing a DBMS. TPC members (that is, the vendors) have little to lose if the intent of the benchmark is, in fact, anything more than a marketing scam.

In my experience, the influence of TPC benchmark numbers is subtle; individuals without knowledge of the TPC see numeric comparisons in sales and marketing literature, conclude subconsciously that the product with the larger number is a better performer, and then remember only that unsubstantiated "fact" when it can have an impact on the purchase decision. DBMS vendors are particularly guilty of fostering the idea that TPC numbers measure DBMS performance, an idea that TPC management acknowledges is arguable. Vendors argue that potential purchasers demand TPC numbers and that there is no better measurement of DBMS performance.

I argue that vendors cannot transfer their responsibilities to users. First, vendors market with TPC numbers — literally defining better performance as higher TPC numbers in advertisements and marketing literature. Second, vendors should tell users that TPC benchmarks don't do the job, rather than protest that there is nothing better. Third, it is vendors who have failed to define better benchmarks. There is one word for insisting on giving customers honest, accurate product information, and for designing products that protect the customer from failure: integrity. Maintaining integrity requires constant vigilance, but in the end, either you have it or you don't.

The recent scandal between Oracle Corp. and the Standish Group regarding benchmarks is shameful. It is a mark of a badly managed standards body when raising concerns (such as those raised by the Standish Group about Oracle's TPC-A benchmark) can result in a member expressing a rebuttal via a lawsuit (such as the lawsuit Oracle brought against the Standish Group for the latter's criticisms).

While it may prompt the TPC to enact policy regarding "benchmark specials," it has really served only to reemphasize the importance of a policy I have practiced for several years. Until the TPC decides that it is more important to act as a provider of useful information than as a broker of marketing and advertising "data," and until it stops hiding behind a policy of secrecy, I must recommend that my clients ignore all TPC benchmark results and inform their colleagues to do likewise. I further recommend that they openly and vehemently protest the use of these numbers by vendors, whether in sales presentations, published articles, or public forums. Unfortunately, government agencies may soon require TPC benchmarks. What a pity. ■

Benchmarking MIDDLEWARE

DAVID MCGOVERAN HAS DESIGNED A NEW DATABASE CONNECTIVITY BENCHMARK (DCB) TO IDENTIFY AND MEASURE MIDDLEWARE CAPABILITIES AND WEAKNESSES.

BY RICHARD FINKELSTEIN

The advent of client/server computing has created a new class of software that sits between the client application and the target database server. This software, which enables connectivity between a variety of tools and database servers, is sometimes referred to as middleware. Evaluating database middleware has been, for the most part, an ambiguous undertaking loaded with risks and uncertainties. Often, products are selected that do not perform well or cannot perform necessary tasks. There are too many dissatisfied customers and failed projects because of middleware software that did not live up to customer expectations. In order to address this situation, David McGoveran of Alternative Technologies has designed and built a new Database Connectivity Benchmark (DCB) that seeks to identify and measure middleware capabilities and weaknesses. (For commentary on the TPC benchmarks, see David McGoveran's article, "Useless Benchmarks: Just Say No!" on page 72.)

Middleware is every bit as complex as, and probably more complex than, the servers and front-end tools that it is connecting. The complexity arises from the fact that middleware has to mediate the differences between two or more heterogeneous software products (front-end tools and back-end SQL database servers, for example) over a variety of communication protocols and mediums. The connection has to be reliable and fast, and must support the functionality that is represented in the client and server software.

The ideal situation would be for the client and server products to interoperate seamlessly with minimal impact from the connectivity software. Currently, most middleware products fall considerably short of this ideal. With the DCB benchmark, it is possible to quantify the difference between the ideal and reality.

The Database Connectivity Benchmark

McGoveran designed the DCB specifically to measure the capabilities of database connectivity products. For this reason, the DCB tests attempt to isolate the performance characteristics of the DBMS so that they do not impact the benchmark results. Unlike standard database benchmarks, for example, DCB databases are defined so that the tables are small and can be cached in memory. This minimizes I/O operations and eliminates this variable during the testing process.

There are two types of tests with the DCB. The first type tests functional limitations and the second tests reliability and performance. Within the first type of tests, DCB tries to determine maximum load characteristics that are often overlooked, even in well-designed DBMS benchmarks. These characteristics include:

- maximum concurrent transactions
- maximum number of accessible databases and tables
- login/logoff rates
- maximum client connections
- capability to recover from DBMS errors and failures, and connection errors
- maximum number of bytes, columns, and rows that can be transmitted

---

Richard Finkelstein is president of Performance Computing Inc., a database-technology consulting company based in Chicago.

- capability to abort in-process transactions
- data-type support
- maximum number of servers that can be supported across multiple nodes
- capability to support SQL Data Definition and Data Control language statements

Ideally, middleware products should be able to support loads that are far greater than the loads you expect to put on the system. Middleware vendors should be able to provide supporting evidence that their products can support the necessary loads in real-life situations (by running the DCB or by providing references). This set of tests provides an excellent feature checklist for understanding and evaluating middleware and DBMS software, even if benchmark tests are not run. Understanding the rationale for the individual DCB functional tests can go a long way toward avoiding disappointments in middleware purchases.

The second type of test attempts to measure performance and reliability when processing a variety of workloads, including:

- batch processing
- report generation
- online transaction processing (OLTP)
- decision support
- ad hoc query
- online complex processing (OLCP)
- mixed transaction loads

The set of workloads is very comprehensive. The goal is to test the middleware product under common transaction-processing conditions — something that other database benchmarks don't often do. The mixed transaction load is designed specifically to measure the interference between individual workloads (such as batch reporting on OLTP). The DCB documentation, however, warns (correctly, in my opinion) that benchmark results are highly dependent on workload, application requirements, system design, and implementation, and that the DCB should not be used as a substitute for critical capacity planning.

The DCB is unique among database benchmarks in the amount of attention it places on data consistency. Most benchmarks either disregard this issue or leave it up to the vendor running the benchmark to choose the level of consistency. Often, transaction benchmarks are run with very low levels of consistency (or none at all) in order to achieve high transaction rates. The DCB mandates performing certain transactions at the highest level of consistency, called "serializability." Serializability guarantees that transactions run simultaneously against the same database will yield correct results no matter how they are interleaved.

Running the Benchmark

Alternative Technologies packages the DCB with several programs that help automate benchmark testing. A database load program is included to assist in populating the database. The database consists of 13 tables that represent a typical order-entry system. The system creates special tables in order to simulate both tables with many columns and tables with wide rows (complex data). The DCB states explicitly that the server must be configured with enough memory to permit caching (in memory) of the entire DCB database. This is done to eliminate the effect of disk I/O on performance measurements.

Alternative Technologies licenses a Driver System, which submits transaction workloads and records response times, free of charge to DCB testers. The Driver System has a standard portion that determines which transactions will be submitted to the DBMS, and an application code portion customized for specific database products. The test sponsor must develop and compile the custom interface. Alternative Technologies provides sample code to help develop the necessary interface. The driver portion reads instructions from a script file that the testing group can customize in order to test special conditions.

DCB's fully automated approach has several advantages over other benchmarks:

- The benchmark is easier to run because Alternative Technologies provides the load and driver systems.
- It is flexible and extensible. You can customize the Driver System scripting language to perform work that is not included or does not pertain to the system under test.
- Results are comparable because each benchmark must use the Driver System and the associated scripts.

Reporting Benchmark Results

The sponsor of the benchmark must issue a Full Disclosure report, which must contain an auditor's attestation letter. The report contains the performance results, including maximum and average response times as well as 90th percentile results (meaning 90 percent of the transactions achieved this response time). The report also includes the number of transactions per second for each workload. The mixed transaction workload, which measures how much work can be done in a given interval, is reported as the total number of transactions (a mixture of many types of transactions) completed during the measurement interval, divided by the number of seconds in the interval.

In order to get a good idea of the range and distribution of response times across all types of transactions, the DCB report contains frequency distribution graphs that plot response time and transactions per second for various levels of throughput. This information is important for understanding the effect that higher transaction rates have on response time. The report also contains comprehensive statistical information (such as transaction-rate standard deviation and variance) in order to evaluate system stability.

The report includes a description of any special settings used to run the test and measurements of system utilization, including memory used, disk I/O activity, and CPU usage. Of special note is the requirement that the test sponsor report the number of times the driver had been used prior to running the benchmark. The intent is to prevent repeated attempts to run a test and report only the ones that succeed. The test sponsor must also provide a detailed list of hardware and software and report a five-year pricing of the entire test configuration.

Conclusions

DCB sets a new high watermark for database benchmarking. The DCB is an exhaustive effort to close loopholes that have plagued other industry benchmarks. Unlike the TPC benchmarks (TPC-A, -B, and -C), which were designed essentially by RDBMS vendors and are highly biased toward showing the product in the best light (by reporting absurdly high transaction rates), the DCB intends to objectively uncover problems and issues in he connectivity products it tests. In other words, Alternative Technologies designed it to protect the customer by providing useable information.

The most important aspect of the DCB is that it does not report simple metrics such as transactions per second or dollars per second. These figures often have no meaning and are easily contrived by vendors. DCB analyzes maximum load capabilities, reports on problems that arise in benchmarks, ensures that high levels of data integrity are supported, and provides detailed performance metrics so that product profiles can be better understood.

In fact, the benchmark is so comprehensive and restrictive, that it may end up that no vendor will run DCB because it may reveal major product weaknesses. This means that it may be up to customers to run DCB themselves. However, if enough customers insist that vendors run the DCB, the DCB may become an industry standard. The DCB specification document itself is an invaluable reference for understanding potential bottlenecks and failure points in database and database connectivity products. I highly recommend it to anyone who is in the process of evaluating middleware or RDBMS software. The DCB will be available to interested parties, in December 1993. There are no fees for using the DCB other than the cost of copying, mailing, and handling. ■

• Alternative Technologies, 13150 Highway 9, Ste. 123, Boulder Creek, CA 95006; 408-338-4621 or fax 408-338-3113.