TANDEMCOMPUTERS

NonStop SQL

**Distributed Relational
Database Management**

**Debit/Credit Benchmark
February 1987**

NonStop SQL Performance

"[The Tandem benchmark has] without question the most complete, publicly released performance documentation to date by any major vendor."

Omri Serlin

Using a relational database management system (RDBMS) with the SQL data language is widely accepted as a more effective way to develop applications, providing more productivity, flexibility, and portability than nonrelational offerings.

Until now, however, no RDBMS has addressed the requirements of on-line transaction processing (OLTP). These requirements include good price/performance, data integrity, continuous operation, and an ability to support applications demanding large capacity.

NonStop SQL™ addresses these problems. Tandem Computers' implementation of the ANSI-standard SQL data language, NonStop SQL, is backed by a powerful, distributed relational database management system. It provides the productivity of the relational model with SQL, but is also capable of meeting the performance demands of critical business applications that require on-line transaction processing.

To substantiate the performance of NonStop SQL, Tandem formally tested it using the industry-standard Debit/Credit Benchmark. This benchmark measures system throughput in transactions per second (tps), a standard measurement in the OLTP industry.

Tandem's goal in implementing the Debit/Credit Benchmark was to develop a useful, realistic, and well-documented appraisal suitable for comparison with other systems. To do this, an independent third party, the highly respected Codd and Date Consulting Group, was invited to audit and verify the benchmark proceedings. Fourteen Tandem engineers and a host of support personnel also participated in the month-long benchmark.

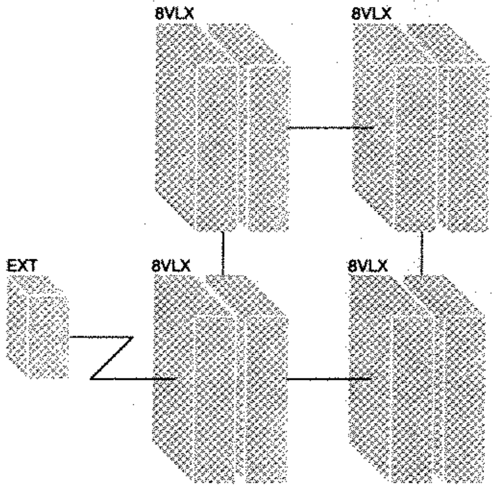
Omri Serlin, a noted analyst of transaction processing systems, commented on the benchmark in the March 1987 issue of his *Fault Tolerant Systems* newsletter (FTSN).

"Unquestionably unprecedented is Tandem's new benchmarking effort. . . . the massive back-up data which Tandem plans to release today will establish a new standard for 'full disclosure.' It follows many of the reporting recommendations published in FTSN-47 in July; and is without question the most complete, publicly released performance documentation to date by any major vendor.

"Hats off to Tandem. May others follow in your footsteps."

Equally impressive were the results of the benchmark. With a response time of two seconds, the throughput achieved on our largest mainframe, the VLX, was 208 transactions per second. The cumulative cost over five years for hardware, software, and maintenance fees was approximately $53,300 for each transaction per second. This price/performance figure is three to five times better than any other SQL product currently available.

For more information on the benchmark and its results, turn to the white paper by Dr. Jim Gray. A reprint of the official audit report from the Codd and Date Consulting Group follows the white paper.



The benchmark configuration

NonStop SQL

Distributed Relational Database Management

Debit/Credit Benchmark
February 1987

Jim Gray

White Paper

"NonStop SQL is the first SQL system to offer the high performance necessary for on-line transaction processing..."

Dr. James Gray is in the Software Development department of Tandem Computers, where he worked on the design and implementation of NonStop SQL. He also served as liaison between Software Development and the Tandem benchmark staff as well as the Codd and Date audit team. Prior to joining Tandem in 1980, he worked at IBM Research on projects including System R, SQL/DS, DB2, IMS-Fast Path, and VM/370 enhancements. Dr. Gray wrote his doctoral dissertation at the University of California, Berkeley, on the theory of precedence parsing. He has written numerous articles, lectured at universities, and reviewed database application designs. He is currently working on heterogeneous databases and is writing a textbook on transaction processing.

Tandem's NonStop SQL™ is an implementation of the ANSI SQL relational database language. In contrast to other SQL implementations, NonStop SQL is designed for on-line transaction processing as well as for information-center use. It delivers high performance cost-effectively and supports distributed data with local autonomy.

This article describes a recent benchmark of NonStop SQL, which demonstrated over 200 transactions per second (tps).

The purpose of the benchmark

The NonStop SQL performance assurance group compared the throughput of ENSCRIBE™, Tandem's file access method, and the new SQL system. The benchmark was done jointly by Software Development in Cupertino, the High Performance Research Center in Frankfurt, and the Benchmark Center in Sunnyvale.

Parts of the benchmark were audited and certified by an independent auditor.

The benchmark demonstrated the following aspects of NonStop SQL:

- NonStop SQL is functional and has been stress-tested for high-volume on-line transaction processing (OLTP) applications.
- NonStop SQL allows distributed data and distributed transactions.
- NonStop SQL runs on small departmental systems as well as on large mainframe systems.
- NonStop SQL demonstrates linear increases in throughput when disks and processors are added using the DYNABUS or the FOX™ fiber-optic ring.
- There is no apparent limit to the transaction throughput of NonStop SQL systems. In particular, there are no bottlenecks in systems running hundreds of transactions per second.
- NonStop SQL performs as well as the record-at-a-time ENSCRIBE system on OLTP applications.
- Both ENSCRIBE and NonStop SQL have impressive price/performance at both low and high transaction volumes.

Editor's note: Text set in bold by Tandem.

```
READ 100 BYTES FROM TERMINAL;
BEGIN WORK;
PERFORM PRESENTATION SERVICES GIVING account__number,
                                     teller__number,
                                     branch__number,
                                     delta;
UPDATE ACCOUNT
        SET balance = balance + :delta
        WHERE account__number = :account__number;
        and balance + :delta>=0
UPDATE TELLER
        SET balance = balance + :delta
        WHERE teller__number = :teller__number;
UPDATE BRANCH
        SET balance = balance + :delta
        WHERE branch__number = :branch__number;
INSERT INTO HISTORY VALUES
   (timestamp,account__number,teller__number,branch__number,delta);
PERFORM PRESENTATION SERVICES;
COMMIT WORK;
WRITE 200 BYTES TO TERMINAL;
```

FIGURE 1.
Profile of the debit/credit transaction.

- The price/performance of Tandem systems is competitive for both departmental systems (NonStop EXT10™) and data center systems (NonStop VLX™).

The definition of throughput and price/performance

The benchmark was based on a standard debit/credit banking application. The article "A Measure of Transaction Processing Power" (*Datamation*, April 1985) defines a standard database, a standard terminal network, and a way to scale them to larger systems. It also describes a standard transaction, called the debit/credit transaction, and specifies how to measure the throughput and price/performance of the resulting system.

Briefly, the database consists of four SQL tables:

- *Account*: a table of bank accounts. Each record is 100 bytes long and holds the account number and balance.
- *Teller*: a table of tellers. Each record is 100 bytes long and holds the teller number and cash position.
- *Branch*: a table of branches. Each record is 100 bytes long and holds the branch number and cash position of all tellers at the branch.
- *History*: an entry sequence table containing a list of all transactions. Each record is 50 bytes long and holds the account, teller, branch, delta, and timestamp.

The transaction, coded in SQL, is shown in Figure 1.

The system is presented with various transaction rates, and the response time is measured over 10-minute windows, creating a response-time curve. (See Figure 2.)

A system that can run one such transaction per second, giving less than one-second response time to 95 percent of the transactions, is called a 1-tps system. The database of a 1-tps system is defined as follows:

100,000	Accounts
100	Tellers
10	Branches
2,590,000	History

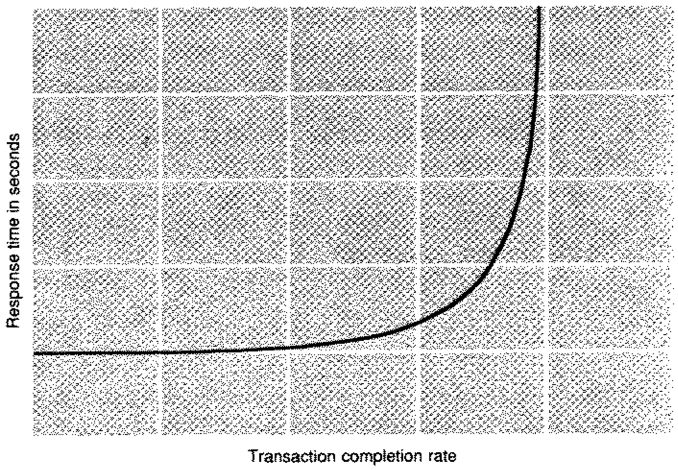


FIGURE 2.
A typical response-time curve showing how response time grows as the transaction load on the system increases.

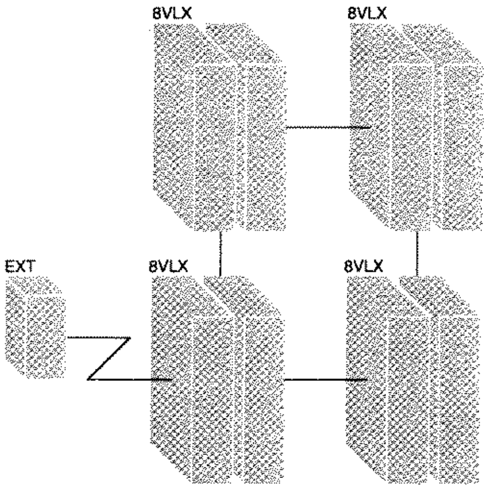


FIGURE 3.
The benchmark configuration. The two-processor EXT10 is connected to one VLX via a 9.6-kilobit line. The 32 VLX processors are divided into four nodes of 8 processors each. The nodes are connected by the high-speed FOX ring. The whole complex looks like a single system to the application programmer. A separate driver system submits transactions to this system via 56-kilobit X.25 lines.

FIGURE 4.
The configuration of each eight-processor VLX node in the FOX ring.

The history table is sized to accommodate 90 days of history records, assuming the average rate is one-third of the peak transaction rate.

The standard also specifies that a 1-tps system has 100 tellers at 10 branches. Each teller thinks for an average of 100 seconds and then submits a transaction. The tellers use block-mode terminals with 10 input and output fields. The input message is 100 bytes, and the output message is 200 bytes. Messages are transmitted via the X.25 communication protocol.

Systems that run more than 1 tps have the database and network scaled linearly. For example, a 100-tps system has a database and teller network 100 times larger.

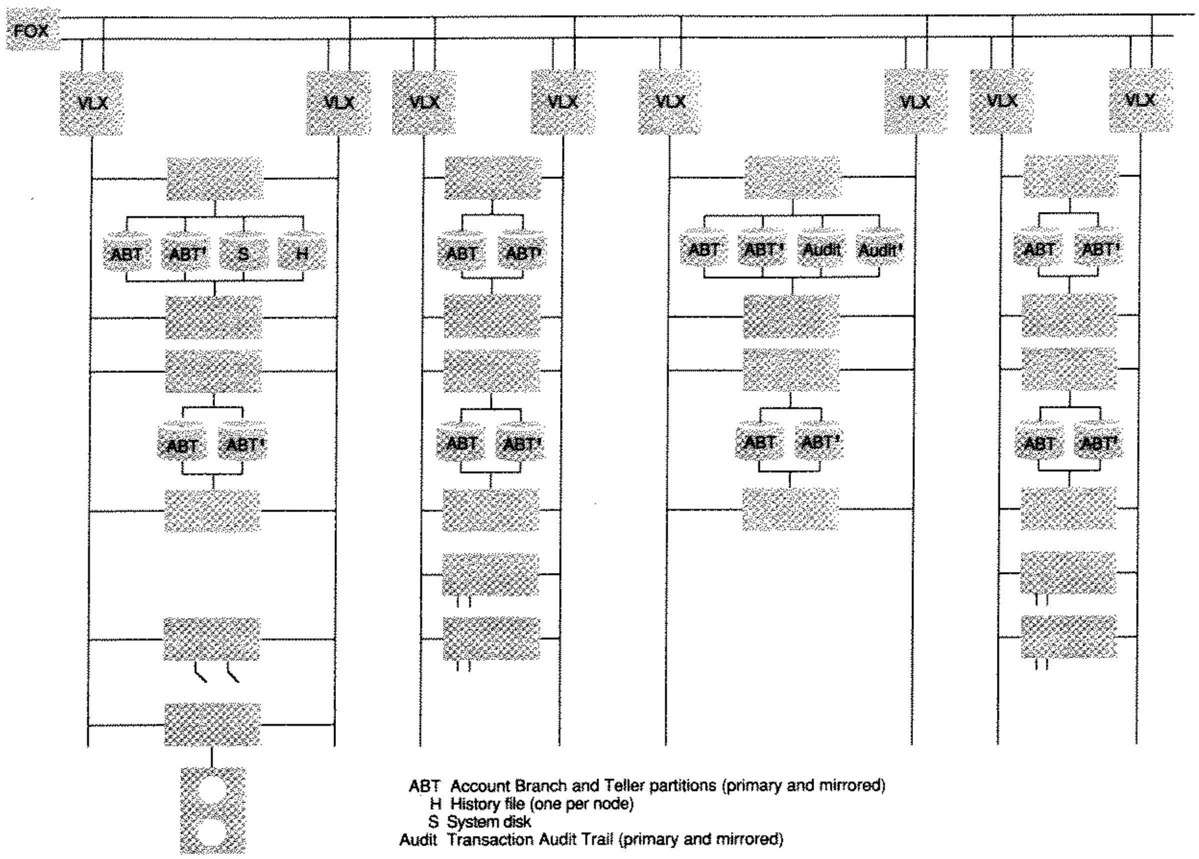
The standard specifies that accounts belong to branches, and tellers belong to branches. If the database is distributed, then 15 percent of the transactions arrive at branches other than the account's home branch. These 15 percent are uniformly distributed among the other branches.

The benchmark requires that the transactions be run with undo/redo transaction protection (abort, autorestart, and rollforward recovery). In addition, it specifies that the transaction log must be duplexed.

Departures from the standard debit/credit transaction

The NonStop SQL benchmark departed from the *Datamation* standard definition in the following ways:

1. Terminals were driven in record mode (Intelligent Device Support) rather than block mode. In effect, this assumes that presentation services are done in the terminal.
2. It was assumed that each branch had a concentrator that multiplexed the teller terminals at the branch. This had the effect of reducing the number of sessions by a factor of 10 for the same transaction rate, when compared to the standard.
3. The measure of tps was the throughput when 90 percent of the transactions get a two-second (or less) response time. The standard measures tps at one second (or less) for 95 percent of the transactions.
4. The SQL system software verified that debits did not cause negative account balances.
5. The system was measured over 10-minute periods, and all transactions for each period were used to compute the response-time curves and consequent tps ratings.
6. Response times were measured within the driver system. The standard specifies that response can be measured at the interface to the service system.
7. All devices were driven by NonStop processes so that no single failure would cause a denial of service.
8. All disks were mirrored (duplexed), as is standard with Tandem equipment.


FOX
VLX
VLX
VLX
VLX
VLX
VLX
VLX
VLX
ABT
ABT'
S
H
ABT
ABT'
ABT
ABT'
Audit
Audit'
ABT
ABT'
ABT
ABT'
ABT
ABT'
ABT
ABT'
ABT
ABT'
ABT Account Branch and Teller partitions (primary and mirrored)
H History file (one per node)
S System disk
Audit Transaction Audit Trail (primary and mirrored)

9. Standard products were used. All applications were written in COBOL85.

Items 1, 2, and 3 tend to create an optimistic tps rating. Items 4 through 9 tend to reduce the system's tps rating.

The benchmark system design

The benchmark hardware comprised 32 VLX processors (32VLX). Each VLX processor had a 5-megabyte-per-second channel, 8 megabytes of main memory, and was rated at about 3 Tandem mips (million instructions per second).

In addition, a NonStop EXT10 system was included in the benchmark hardware to demonstrate scaleability. The EXT10 is Tandem's smallest available NonStop system. It consists of two processors, each with 4 megabytes of main memory and four disks. The EXT10 processors together are approximately half as powerful as a VLX CPU. Thus, the complex was sized as a 32.5 processor VLX system. (See Figure 3.)

The VLX processors were divided into four nodes of eight VLX CPUs each. Each node connected its eight processors via dual 20-megabyte-per-second DYNABUS interprocessor buses, and the nodes were interconnected via FOX, a dual, bidirectional fiber-optic ring. (See Figure 4.)

To model a distributed node, the EXT10 was connected to the VLX processors with a 9.6-kilobit communication line.

The GUARDIAN 90XF™ system made the entire 32VLX plus EXT10 configuration appear to be a single system with location transparency.

The database and transaction load were partitioned into 33 parts. Based on preliminary tests, each VLX processor would process less than 8 tps, and the EXT10 could process 4 tps. All sizings were based on these preliminary measurements.

FIGURE 5.
Network and database sizes for the 208-tps benchmark system.

	Database	
Network	Records	Size
25,600 Tellers	25,600 Tellers	2.6 Mbytes
2560 Branches	2560 Branches	0.3 Mbytes
	25,600,000 Accounts	2.6 Gbytes
	54,000,000,000 History	27.0 Gbytes

FIGURE 6.
The SQL statement used to create the account file partitioned among 33 disks.

```
CREATE TABLE account (number   PIC 9(12),
                      balance  PIC S9(11)V(2),
                      ...
                      KEY number
                      )
                      PARTITION $vlx2 FIRST KEY      800000
                      PARTITION $vlx3 FIRST KEY     1600000
                      ...
                      PARTITION $vlx32 FIRST KEY 24800000
                      PARTITION $ext    FIRST KEY 25600000;
```

Each transaction input message was 100 bytes long; the reply was 200 bytes long. With X.25 overheads, this translates to 340 bytes in all. At 8 tps, this is 22 kilobits per second. A 56-kilobit line can handle this load quite comfortably. Therefore, each VLX processor was configured with a 56-kilobit line, and the EXT10 was configured with a single 56-kilobit line to the driver system.

The database was sized as follows. Each eight-processor VLX (8VLX) node had:

- A mirrored disk to store the programs and the transaction log (audit trail)
- A mirrored disk dedicated to the history table. In the benchmark, only one volume was configured, but in pricing the system, the 90-day history file of each node was sized at 6.7 gigabytes (14 mirrored volumes).

According to the standard, 8 tps implies:

80 branches	8 kilobytes
800 tellers	80 kilobytes
800,000 accounts	80 megabytes

These records, along with their indexes and some slack, fit comfortably on Tandem's smallest disk. Therefore, each VLX processor was given a mirrored disk volume to hold its part of the account-branch-teller (ABT) tables. A 1.6-megabyte disk cache per mirrored disk partition was sufficient to keep all branches, tellers, and the account index pages resident in main memory.

Figure 5 shows the overall system configuration. The teller, branch, and account files were mirrored, but for lack of hardware, the history file was not mirrored.

The SQL tables were defined to be partitioned by the appropriate key values. The approximate syntax for creating a single account table with 33 partitions is shown in Figure 6.

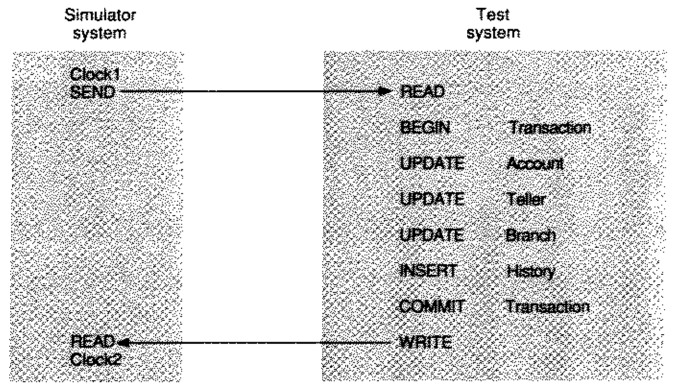
NonStop SQL hides this partitioning from the application programmer. The branch and teller files were partitioned in a similar way.

NonStop SQL imposes no practical limit on the number of partitions a table may have. Because ENSCRIBE is limited to a maximum of 16 partitions, the ENSCRIBE database and ENSCRIBE benchmark were limited to 16 VLX processors.

In the Tandem system, terminal control and presentation services are handled by a process called the Terminal Control Program (TCP). It is the logical equivalent of IMS/DC or CICS. Since each CPU was sized at 8 tps, each CPU supported 800 different teller records at 80 branches. It was assumed that each branch had a concentrator that multiplexed the teller terminals at the branch. This had the effect of reducing the number of sessions by a factor of ten for the same transaction rate, when compared to the standard. It was decided to configure 32 branches (320 tellers) per TCP which resulted in approximately 2.5 TCPs per VLX CPU.

Table 1.
Benchmark hardware configuration.

Driver system	Lines	Processors	Storage
24 MIPS	33*56 Kbyte X.25	32VLX + EXT10 100 MIPS 264 Mbytes	86 disks 20 Gbytes(+ 27 Gbytes history when pricing)



Measured response time = Clock2 - Clock1

Note: The standard specifies the response time as the elapsed time between the last bit entering the test system until the time the first bit leaves the system.

FIGURE 7.
The driver system submits messages via X.25 and measures the transaction completion rate and response time.

To avoid queueing on database servers at 8 tps with a two-second average response time, 20 debit/credit servers were configured per CPU (20 ~ 2*8). The code and data for the TCPs were approximately 0.5 megabytes per VLX CPU. The ENSCRIBE servers consumed 0.25 megabytes per VLX. The NonStop SQL servers used approximately 10 kilobytes more memory—altogether 20 NonStop SQL servers used about 0.5 megabytes per VLX.

The EXT10 system was sized at half of a VLX processor. The programs and audit trail were on one mirrored disk and the database resided on a second mirrored disk.

A second complex of 12 NonStop TXP™ processors simulated the network of 2,560 branches (25,600 tellers) by submitting transactions via X.25, using modem eliminators to connect the X.25 lines. Typically, transactions were submitted to each VLX line at an average rate between 4 tps and 8 tps, with exponentially distributed interarrival times. The EXT10 was driven at about 4 tps.

Each transaction message named an account, branch, teller, and debit amount. As specified by the standard, in 85 percent of the cases, the account and branch were local; in 15 percent of the cases the account was in a branch other than the branch and teller of this transaction. Some of these "nonlocal" transactions were in branches at the same node, but most went to other nodes of the network. For example, when the system was running at over 200 tps, the EXT10 might originate or receive 1 distributed transaction per second, while each VLX node might originate or receive 15 distributed transactions per second.

The driver system measured response time as the time between the send and the completion of the receive in the driver system. (See Figure 7.)

The hardware configuration is summarized in Table 1. Assembling this hardware was difficult. Because the VLX had a backlog of orders, the equipment was only available for a limited time. The hardware had to be assembled, benchmarked, and disassembled within a 50-day time frame. Fortunately, the equipment was installed on schedule. There were no hardware errors during the benchmark, and no critical software errors were discovered in the SQL product itself. The only hardware problem was a double power failure at the facility early in the benchmark.

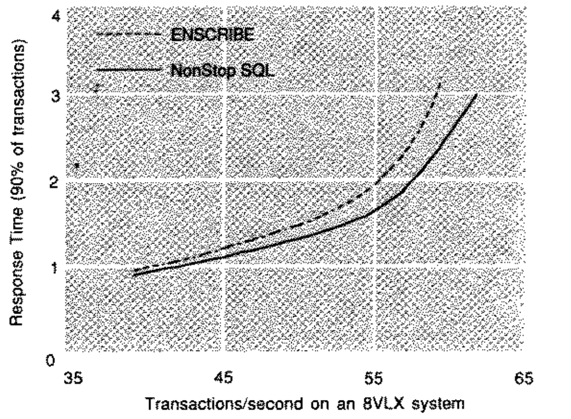
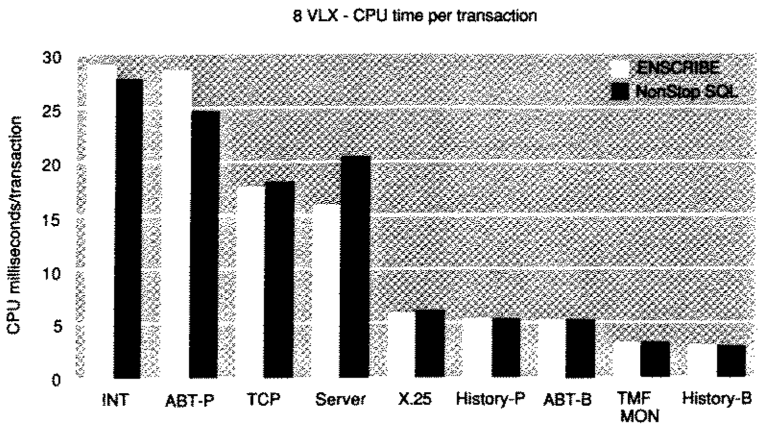


FIGURE 8.
The response-time curves for NonStop SQL and Tandem's record-at-a-time ENSCRIBE system running on an eight-processor VLX system. Notice that NonStop SQL has slightly better performance than ENSCRIBE.

FIGURE 9.
A detailed breakdown of CPU time spent by each component of the debit/credit transaction on an 8VLX system showing both the ENSCRIBE and SQL costs. INT signifies interrupt handling and the message system. ABT refers to the account, branch, and teller disk servers. The primary ABT and backup (process pair) disk processes are separated. The TCP does presentation services and terminal handling. X.25 does the physical line handling for the input and output messages. The disk server storing the history table is shown separately. Lastly, the transaction commit coordinator is represented by TMF MON.



The experiments

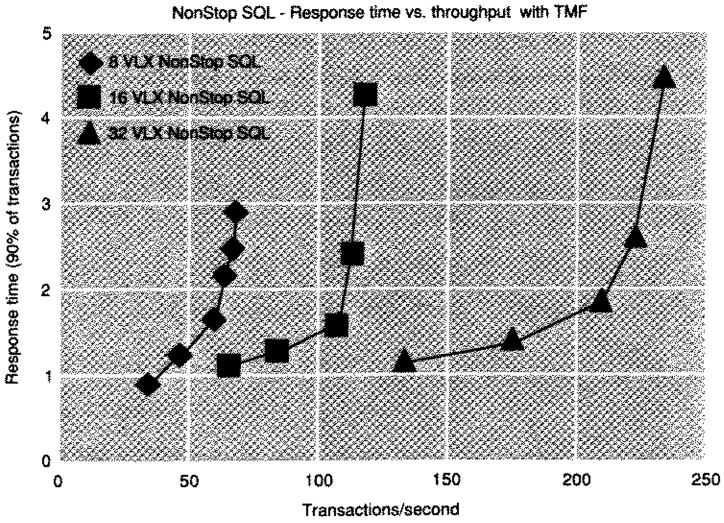
The benchmark measured the ability to grow a NonStop VLX system from eight VLX processors to 32 processors using the FOX fiber-optic ring. In addition, the benchmark demonstrated that NonStop SQL runs on Tandem's low-end EXT10 system attached to the VLX processors via a 9.6-kilobit line. The EXT10 had a proportionate part of the database and sent and received distributed transactions (15 percent).

First, the throughput of a single, 8VLX system was measured for both ENSCRIBE and NonStop SQL. Then, a pair of 8VLX systems were connected via FOX and their performance was measured for NonStop SQL. Finally, the four-node, 32-processor, FOX connected network with attached EXT10 was measured. Because ENSCRIBE is limited to 16 partitions per file, the last experiment was done only for NonStop SQL.

Figure 8 shows the response-time curves for the NonStop SQL and ENSCRIBE tests on an eight-processor VLX system. The curves show that for the same response time, NonStop SQL gives slightly better throughput and, similarly, that NonStop SQL gives better response time for a fixed throughput.

Each transaction generated one physical read and two physical writes of the account file (one mirrored write). Disk reads and writes of the account, branch, and history files are amortized over many transactions. In addition, each transaction contributes about 0.4 physical log I/O per transaction, because group commit batches about five transactions per audit flush.[1] A detailed breakdown of the CPU utilization by function is shown in Figure 9.

FIGURE 10.
The tps rating of VLX processors for NonStop SQL. At two-second (or less) response time for 90% of the transactions, the system was running at a sustained rate of 208 SQL tps.



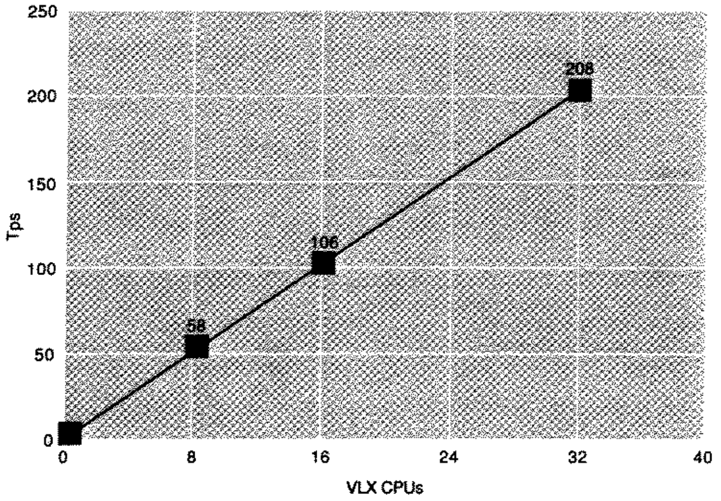


FIGURE 11.
The tps rating of NonStop SQL running the Debit/Credit Benchmark on VLX processors. On 32 VLX processors, the system processed 208 tps.

These experiments were repeated for 16 VLX processors. Then the NonStop SQL experiments were repeated for 32 VLX processors. (ENSCRIBE is limited to 16 partitions.) The response times were plotted and are documented in the Auditor's Report (Sawyer, 1987). The resulting throughput curves are shown in Figure 10.

These experiments were audited by The Codd and Date Consulting Group,[2] which verified that:

- The transaction was correctly implemented
- The database was sized properly
- 15 percent of the transactions were interbranch
- Transactions were protected by a dual undo/redo log
- The response time was measured correctly
- The measured response-time curves matched the system
- NonStop SQL and ENSCRIBE had comparable performance.

Based on early measurements, the system was expected to perform about 8 tps per CPU and have linear growth from 8 to 32 CPUs, i.e., the 32-processor system would handle approximately 256 tps. In fact, the 8VLX system did 7.2 tps per VLX CPU, and there was a 10 percent dropoff as the system was scaled to 32 processors. This is shown in Figure 11. The dropoff is due to the increased cost of network distributed transactions. When transactions do work at multiple nodes, they cost extra instructions and I/O traffic. This extra cost implies a reduction in throughput overall. In spite of this, the tps curves are linear; however, the slope is 0.9 rather than 1.

In addition, NonStop SQL uses fewer CPU instructions in this benchmark than ENSCRIBE. Consequently, NonStop SQL has a slightly higher transaction throughput. Ignoring response time, peak NonStop SQL throughput was 229 tps.

Given this linear throughput vs. hardware, it is possible to quote the price/performance of the system in terms of the price/performance of a single processor. The five-year cost per transaction is approximately the same for a small system as for a large one. The dominant issue is which processor line the user selects. Generally, newer systems are less expensive.[3]

Why is NonStop SQL so fast and powerful?

Historically, SQL has been confined to information center environments and to low-end systems where programmer productivity is more important than system performance. In these environments, the power of the SQL language and the relational model compensated for the lackluster performance of most relational systems.

[1] *Group commit* means that when a number of transactions are complete, the log file is all written to disk together. (This is also called "piggybacking," or "boxcaring.") Even though the commit records are grouped, they are still atomic in nature, i.e., separable by transaction.

[2] Founded by E.F. Codd, the originator of the relational model, and C.J. Date, a leading author and lecturer on relational technology. The Codd and Date Consulting Group is considered the ultimate source for relational product education, evaluation, and consulting.

[3] More detailed pricing information is available in *The NonStop SQL Benchmark Workbook.*

FIGURE 12.
(a) Comparison of a COBOL record-at-a-time update, which sends a read followed by a write, and the SQL update, which sends a single update message. This message savings helps explain the performance advantage of NonStop SQL compared to ENSCRIBE.

COBOL and ENSCRIBE

```
START branch KEY IS EQUAL TO branch-number
READ branch RECORD with LOCK
ADD delta TO balance OF branch
REWRITE branch
```

NonStop SQL

```
UPDATE branch
 SET      balance = balance + :delta
 WHERE number = :branch-number
 AND      balance + :delta> =0
```

NonStop SQL is the first SQL system to offer the high performance necessary for on-line transaction processing. As demonstrated by the benchmarks, it has performance comparable to Tandem's record-at-a-time ENSCRIBE system. In addition, NonStop SQL is a distributed relational system; other systems do not offer full-function distributed data or distributed execution.

There are several reasons for the success of this benchmark. First, NonStop SQL benefited from the experience of its predecessors. The developers had collectively worked on System R, SQL/DS, DB2, R*, IDM, ENCOMPASS™, Esvel, Wang VS, and several other systems.

A second benefit was the close cooperation between the Database group and the Operating System and Languages group. Tandem is a transaction processing company; the operating system is geared for efficient processing of distributed transactions. NonStop SQL exploits the transaction mechanism of the GUARDIAN 90XF system code to easily and efficiently get transaction-protected distributed execution based on remote procedure calls.

Perhaps the most significant advantage of NonStop SQL is the SQL language itself, especially when contrasting the record-at-a-time interface of ENSCRIBE, DL/1, or DBTG with the set-oriented interface of SQL. Figure 12a shows sample UPDATE statements in the debit/credit transaction.

In the case of ENSCRIBE, a message is sent to the disk process to get the designated branch record. The record is then returned to the program, which examines it, alters it, and then returns it to the database. The result is actually three messages and a lot of data movement. (See Figure 12b.)

NonStop SQL sends a single message to the disk process requesting an update of the appropriate account by the appropriate amount. The disk process applies this update and returns a status message to the caller. The result is half as many messages and one-quarter the number of message bytes.

In general, NonStop SQL subcontracts single-variable queries to remote servers. This allows the NonStop SQL disk process to act as a database machine that performs updates and deletes and filters data, returning only the desired rows and columns of a table.

The examples shown in Figure 12 illustrate the synergy between SQL and distributed database systems. It has long been felt that SQL would be a good basis for distributed database systems. In every benchmark the message savings of NonStop SQL have compensated for the extra work the system must perform in order to implement the more complex semantics of the SQL language. For example, NonStop SQL is nearly twice as fast as ENFORM™, Tandem's report writer for ENSCRIBE files, on the Wisconsin Benchmark (Bitton). Most of the increased speed is due to the close integration of NonStop SQL with the operating system.

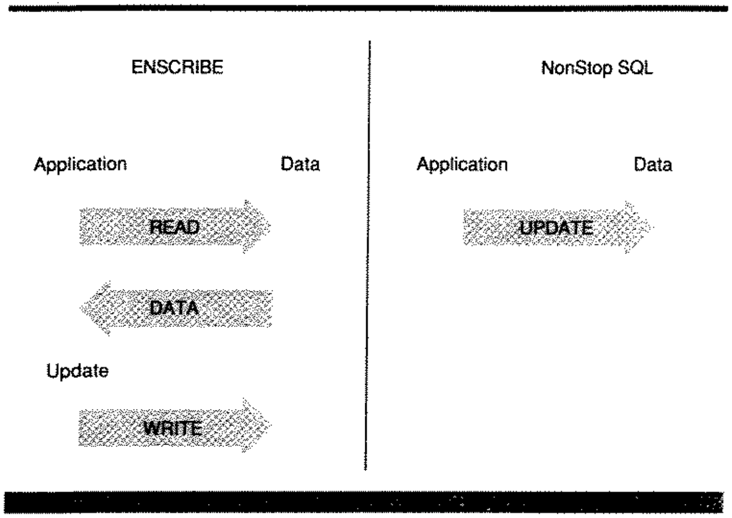


Figure 12.
(b) SQL can update or filter data at the source, whereas record-at-a-time interfaces must fetch the data to the application program. In this way, SQL saves messages and CPU cycles as compared to a conventional data management system.

Even in the best circumstances, a system might bottleneck on certain resources. For example, most transaction processing systems bottleneck at 30 tps because they have not implemented group commit, which batches commit records (Gawlick, 1985). The fact that NonStop SQL was benchmarked using 16 processors on a single DYNABUS, and at over 200 tps without any bottlenecks, suggests that there are no obvious bottlenecks in the system.

Conclusion

Traditionally, relational systems have had a reputation for poor performance. NonStop SQL, however, performs comparably to traditional, record-at-a-time nonrelational systems. Running over 230 tps on a 32 processor VLX complex and an EXT10 demonstrates that NonStop SQL is functional, distributed, and scaleable. In addition, it has good price/performance and flat price/performance from the low end to the high end. There are no performance limits and no performance penalties for NonStop SQL.

This article was originally written for the *Tandem Systems Review*. It is scheduled to appear (in a different format) in the February 1988 issue (Vol. 4, No. 1).

REFERENCES
Anon, et al. 1985. A Measure of Transaction Processing Power. *Datamation*. Vol. 31, No. 7.

Bitton, D., et al. 1983. Benchmarking Database Systems: A Systemic Approach. Proc. 9th VLDB. IEEE Press.

The NonStop SQL Benchmark Workbook. Part no. 84160. Tandem Computers Incorporated. (Available through Tandem sales representatives.)

Gawlick, D. 1985. Processing Hot Spots in High-Performance Systems. IEEE COMPCON '85.

Codd and Date
Consulting Group

Mr. Frank Clugage
Tandem Computers Incorporated
19191 Vallco Parkway
Cupertino CA 95014

March 6, 1987

Mr. Clugage:

The attached report describes the portions of the Debit/Credit benchmark audited by Codd and Date Consulting Group.

The 32-processor VLX system ran the benchmark in excess of 200 transactions per second for a period of 15 minutes.

The following additional attributes of the benchmark were verified:

» The transaction is correctly implemented

» The database is properly sized

» The programs are unaware of data distribution

» The percentage of inter-branch transactions is correct

» The response time is correctly measured

» The inter-arrival times are exponentially distributed

» The recovery log is mirrored

» Database updates are locked during a transaction

» The performance is linear with system size

» The application is implemented in COBOL and SQL

» The network can mix small and large systems.

Respectfully yours,

Thomas H Sawyer

Thomas H. Sawyer

Senior Consultant

NonStop SQL

Distributed Relational Database Management

Debit/Credit Benchmark
February 1987

Codd and Date Consulting Group

Audit Report

"The 32-processor VLX system ran the benchmark in excess of 200 transactions per second for a period of 15 minutes."

Dr. E. F. Codd, the originator of the relational model, remains a leading authority on relational database technology. A former IBM fellow, the recipient of the 1981 ACM Turing Award, and an elected member of the National Academy of Engineering, Dr. Codd is now the senior vice president and chief scientist of the Codd and Date Consulting Group.

Chris Date, one of the most widely read and best known author/lecturers in the field of computing, has been involved in the relational database field since 1970. His book, *An Introduction to Database Systems*, Volume I, Fourth Edition, serves as a standard text on database technology. After a successful 16-year career with IBM, he is now the executive vice president of the Codd and Date Consulting Group.

Benchmark synopsis

The debit/credit transaction is a stylized automatic teller transaction. The process consists of updating the account balance of the teller user. The balance of the teller and branch are also updated. A history record is inserted to complete the database portion of the transaction.

The terminal portion of the transaction consists of receiving a 100-byte record from the automatic teller and returning a 200-byte record at transaction completion.

At least 15 percent of the accounts affected are in a different branch than the teller processing the transaction.

For a complete description of the debit/credit transaction, see "A Measure of Transaction Processing Power" by Anon., et al., contained in the Auditor's Notebook.

Observed results

Two hardware configurations were audited, each of which ran the debit/credit transaction implemented in COBOL and SQL. Both configurations delivered the performance documented elsewhere in this report. The verification section details the auditing techniques.

The larger hardware configuration consisted of four 8-processor VLX systems connected via a FOX ring. One VLX system was also connected to a remote EXT10 via a 9.6-kilobit transmission line. Transactions were submitted directly to each system (including the EXT10).

The correct percentage of these were interbranch. The EXT10 also participated in the interbranch transactions at a rate commensurate with the size of its database.

Benchmark implementation

The verification procedures are dependent on the Tandem implementation of the benchmark. Tandem implemented the benchmark with two sets of hardware—a driver system and the system to be measured.

The driver system (a 10-processor TXP system) was connected to the measured system via 56-kilobit transmission lines. There was one line for each processor in the measured system. A transaction-submitting program (the Driver Program) on the driver system simulated an automated teller network.

All measurements were taken, recorded, and summarized on the driver system. The measured system contained two application programs.

Requestor Program

One program (the Requestor Program) received the transaction from the communication software, reformatted the transaction data, and assigned it to a copy of the other program (the Server Program).

Server Program

The Server Program performed the database updates and returned a confirmation to the Requestor Program. The latter program then sent a response to the driver system to complete the transaction.

Steps involved in a timing run

Script preparation

The data for each transaction is prepared by a script-generation program. Each transaction record contains the account, branch, teller, and amount involved.

The script generator uses a random number generator to determine the account number. This number is then used to determine the branch and teller. The random number generator is used a second time to determine if this transaction should be interbranch. If so, the random number generator is used again to pick an account that is not in the branch.

Transaction timing

Transaction timing is performed by the driver program as part of submitting transactions to the measured system. The transaction is time-stamped when it is sent to the X.25 component on the driver machine. A second time-stamp is taken when the X.25 component returns the response to the driver software.

The response time is the difference of the two time-stamps. Note that the time-stamps include all X.25 processing and all transmission time in the response time. The standard benchmark response-time requirement only counts the time spent in the measured system. Thus, the true response times are actually less than reported.

When the transaction completes, the driver calculates the interarrival rate of the next transaction, relative to when it was submitted. If the interarrival time was exceeded by the response time, the next transaction is submitted immediately. If there is any remaining "think time," the transaction is queued until the think time elapses.

Tandem chose an exponential interarrival rate. This rate is the most difficult for transaction processing systems to accommodate. The easiest is a constant interarrival rate—i.e., each terminal submits one transaction every 10 seconds with no variance.

The Driver Program writes a report record showing the response time, the calculated interarrival time of the next message, and the delay factor if the response time exceeds the interarrival time.

Report preparation

After the transaction submission portion of the run is stopped, the report files from the driver system are collected and summarized. This summarization produces the transaction-rate graph and interarrival-rate graphs.

The interarrival-rate graph shows the theoretical and the actual interarrival rate for each run. The theoretical rate is usually not met for extremely short interarrival times—the actual response times are longer. Where the computed times exceed the response time, the distribution takes on the correct shape.

The calculations and report logic were verified by inspecting the source code. A further check was made by spot-checking the disassembled code from the execution module with the listing.

Verification

Interbranch transactions

The required interbank transaction ratio of 15 percent was verified two ways. The script-generation code was inspected to validate the percentage generated, and the output was spot-checked to confirm the percentage. The spot check yielded 16 percent.

Response time

Response time was verified two ways. The driver and reporter codes were inspected for correctness. The driver code was modified to take additional transactions directly submitted by the auditor. During the transaction-submission portion of the run, direct transactions were submitted. These transactions updated accounts on each of the nodes in the network. Upon receipt of the response, the computed response time was noted and later compared with the response times generated by the reports.

Linear transaction rate

The benchmark attempted to show an absolutely linear performance improvement as nodes were added to the system. The actual improvement is very close to linear, but shows a growing divergence from true linearity as nodes are added. The auditor ascribes most of this falloff to the effects of the interbranch transactions.

In the benchmarked configuration, each VLX processor directly dealt with eight branches; a VLX system had eight processors. Thus in an eight-processor system, all interbranch transactions would have their recovery scope contained on one system. The communication within a VLX system uses dual high-speed buses (24 megabytes per second) and has no need to coordinate recovery scope with any other node on the system.

As nodes are added to the system, the percentage of transactions requiring intersystem coordination increases. For example, a system made up of two 8-processor VLXs would have one-half of the interbranch transactions spread across the network. As the number of nodes rises, the percentage increases until it approaches 100 percent of all interbranch transactions or 15 percent of all transactions.

Indeed, the extrapolation of the 8-processor transaction rate to 16- and 32-processor systems predicts 116 and 232 tps. The interpolated rates were 106 and 208. These are 8 percent and 10 percent below linearity. The percentage of internode, interbranch transactions for these two configurations is 7.5 percent and 11 percent.

Distributed database capability

The benchmark demonstrated three aspects of distributed database support: program transparency, distributed updates, and asymmetric node support.

- *Program transparency*
  This feature allows applications to be coded as though all data is at one location. The system takes care of processing any data remote to the node executing the transaction.

  The Tandem system permits spreading of a table across multiple systems based on the values of one column. This is commonly referred to as partitioning. For the benchmark, the accounts database was spread over five network nodes—one EXT10 and four VLX systems. I inspected the application program (Server); there is no knowledge of this partitioning in the program.
- *Distributed updates*
  This feature coordinates updates that occur on more than one node for one transaction. This means the system must be prepared to abort all work done by a transaction on all nodes where it occurs.

  The interbranch transactions produce multiple node updates for 11 percent of the interbranch transactions.

  Additional tests were performed using the interactive SQL interface (SQLCI) where all updates were on a node different from the one processing the transaction.
- *Asymmetric node support*
  This feature allows processors of different power to participate in the same transaction. The EXT10 participated in both local and interbranch transactions even though the two-processor EXT10 has less than half the power of a single VLX processor. It should be noted that the debit/credit transaction is not a heavy-duty test, since it does not have much complex processing.

Duplexed or mirrored log file

The benchmark requires the log file to be duplexed. That is, that the system should be capable of losing one of the log files and still be able to recover or abort transactions.

This capability was demonstrated in the following manner. SQLCI was used to start a transaction and update an account. The power was turned off for one of the log disks. The transaction was aborted. The account balance was queried and shown to be restored to the balance found prior to the start of the transaction.

Additional checks performed

Several additional checks were performed to ensure the validity of the benchmark.

- *Database locking*
  To prove that record locking was actually occurring, the following test was performed. SQLCI was used to start a transaction. An account balance was updated and queried to demonstrate the change. A second transaction was started on another terminal. The same account was queried. The balance was not returned to the second transaction, which was timed out after waiting a decent interval. If there were no locking mechanism, the balance would have been returned to the second transaction even though the first had not completed.

- *Transaction message size*
  The benchmark calls for an input message of 100 bytes and a 200-byte confirmation. This was verified in the following manner. The driver code was inspected to ensure it was sending a 100-byte message. The code was also inspected to verify that it calculated the length of the response correctly. The auditor transaction-submission code was inspected to verify the message length response was correct.

  The length of the response was spot-checked during the submission of transactions during the test. Several additional transactions were submitted with invalid account numbers. These elicited error responses of a different length than the normal transactions.
- *Hardware configuration*
  Several checks were made to validate the hardware configuration.

  The nodes on a Tandem system are connected with a fiber-optic ring (FOX) when high transmission rates are sought. To verify that only the benchmark systems were on the ring, the tiles in the machine room were removed and the cabling traced.

  All disks not directly needed for the benchmark were turned off.

  All systems not used in the benchmark were inspected to ensure no FOX connection.

  The line speed setting between the measured system and the EXT10 was verified at 9.6 kilobits per second.
- *Database sizing*
  The Tandem benchmark system was configured for a maximum transaction rate of 256 tps. This rate requires 25.6 million account records of 100 bytes each. In addition, the EXT10 required 400,000 account records.

  The above was verified by inspecting the data definition statements for the account, branch, and teller tables. Additionally, accounts in each of the nodes were updated via transactions submitted during the timing portion. These accounts were then queried using SQLCI to verify the updates had taken place.
- *Interarrival rate*
  To verify the random interarrival rate, code was added to the reporting program to plot the theoretical and actual distribution. This code was inspected to ensure correctness.
- *Reverified all code each day*
  The timing runs were performed over several days. To ensure the same software was used each day, the auditor kept a tape of the executable modules for the driver system. The contents of this tape were compared to the programs used for each run.

  The code on the measured machine was inspected prior to each day's runs.
- *Verified use of generated scripts*
  After the script files were generated, one of them was picked, and several of the transactions were altered to update balances not otherwise accessed. These transactions were scattered through the script to ensure the entire script was used. After a timing run, the balances of these accounts were verified.

Conformance to Debit/Credit Benchmark

Deficiencies

- *Terminal sizing*
  The benchmark calls for a configuration of 10 terminals for every branch. Each terminal has a 100-second think time. Tandem reduced the number of terminals by 10 and reduced the think time by 10 seconds. This change retained the transaction rate but does have the effect of reducing the amount of memory needed to support the terminal.
- *Response-time graph*
  The benchmark states that 95 percent of all transactions must complete in one second. Tandem uses two seconds for 90 percent of all transactions.
- *History database*
  The history database did not have physical disks allocated to cover the 90-day requirement. This is a minor matter, since this database is updated at the end. The priced configuration should have the correct amount of disk included.

Extensions

- *Mirrored disks*
  All data is written to its primary location and its mirrored location. This doubles the number of disk I/Os for each transaction.
- *Message interarrival rate*
  The benchmark does not specify the pattern of time between messages. The easiest is a constant time. For this benchmark that would be a rate of one message every 10 seconds per terminal. Tandem used an interarrival rate that was exponential but had a mean of 10 seconds per terminal. This rate is a worst-case assumption.
- *Response-time measurement*
  According to the benchmark definition, response time starts when the last bit of the transaction is received in the measured machine and ends when the first bit is sent out on the transmission line. It must include all time spent in the measured machine.

  Tandem measured the response time from the time the message was accepted by the X.25 component on the driver side of the communication line until the last byte of the response had been processed by the X.25 component on the driver machine. Thus, all message processing and queueing that occurred on the driver machine and all the line time were added to the response time.
- *Implementation language*
  Tandem implemented all application-dependent code in COBOL85. The database processing was implemented in SQL rather than a machine-specific access mechanism.
- *Account balance testing*
  The benchmark makes no requirement to test the account balance for overdrafts. Tandem included an overdraft protection check in the transaction. This made the transaction a bit more realistic.

Conformance to ANSI-standard SQL

Three exceptions were noted.

The WHERE clause that specifies which teller or which branch is updated references a field called "syskey." This field is not a data value in the row; it is the relative record number—in effect a direct reference. This usage removes the access transparency normally gained by SQL. That is, the program is dependent on the underlying implementation. The same effect could be obtained by enhancing the CREATE TABLE statement to include the information that the branch number is equal to syskey.

A second exception to the ANSI standard is the use of COBOL picture clauses in the column definitions under CREATE TABLE.

All table names in the application SQL statements are preceded with equal signs (" = "). This is a Tandem idiosyncrasy that concerns the mapping of table names to data files. Unfortunately, it makes the programs nonportable.

The CREATE TABLE statement contains one extension—the information needed to partition the table.

TANDEMCOMPUTERS

Tandem Computers Incorporated
World Headquarters, Cupertino,
California, USA
Offices throughout the United States,
Canada, Mexico, Europe, the United
Kingdom, Hong Kong, Singapore, Japan,
Australia, and New Zealand.
Distributors in Argentina, Colombia,
Finland, Korea, Malaysia, the Middle
East, Peru, Taiwan, Thailand, and
Venezuela.

Printed in USA P70407-25M-103996-1187