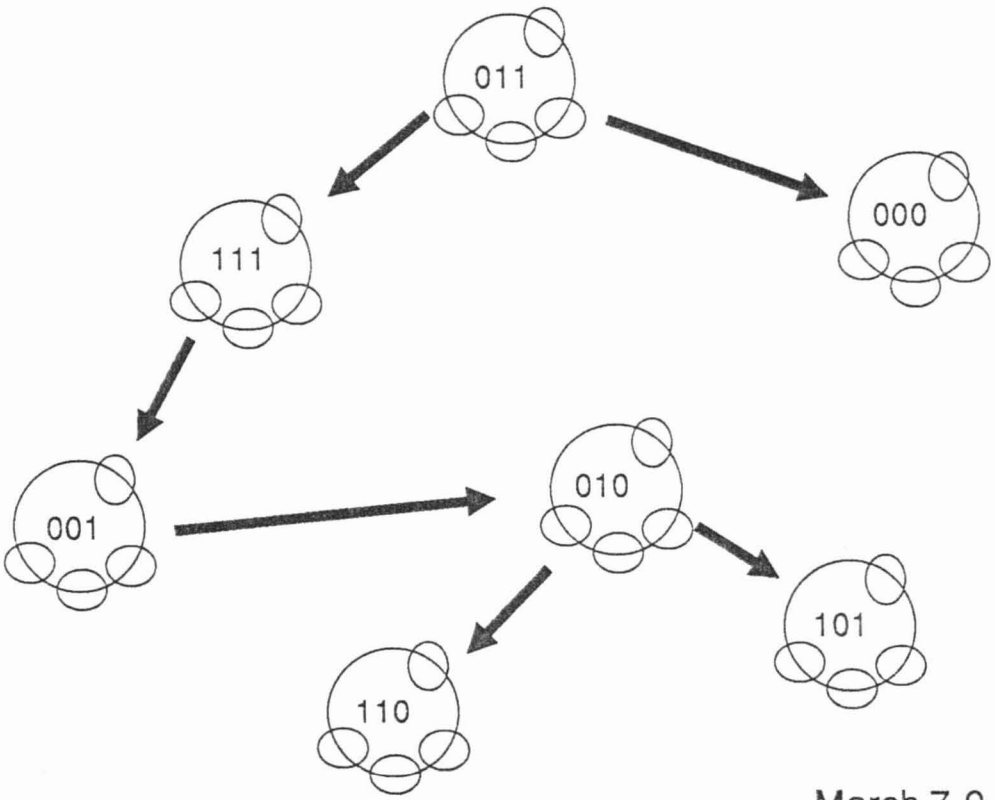
Databases and Parallel Architectures

Proceedings

PARBASE-90

International Conference on Databases, Parallel Architectures, and Their Applications

Edited by N. Rishe, S. Navathe, and D. Tal



March 7-9, 1990
Miami Beach, Florida

Sponsored by
Florida International University
in cooperation with IEEE and Euromicro

IEEE Computer Society Press

Institute of Electrical and Electronics Engineers, Inc.

Structured Evaluation of Database Management Systems

Gary L. Schaps

Cordis Corporation
Miami, Florida 33102

Naphtali Rishe

Florida International University
Miami, Florida 33199

ABSTRACT

This paper presents an analytical framework for evaluating database management systems. Four leading commercial systems are reviewed in this context.

Selecting a database management system for building a large scale, performance-critical application is a difficult and complicated task. Generic comparisons may be insufficient because the capabilities they test are only a small portion of your application[20]. Although it may be tempting to choose the fastest product on the basis of one of these tests, a better strategy is to perform your own benchmarks. But a benchmark is expensive and time consuming. One has to learn to use the DBMS before being able to build the prototype. In addition, it has to reflect the performance critical portion of the application, and it has to be tested under realistic conditions by the maximum number of simultaneous users. With these facts in mind, generic comparisons of products in the preliminary selection phase may offer some appeal. An alternative approach is to structure a preliminary selection on the basis of the DBMS design, select the design which offers the best potential and construct a benchmark based on your application. The goal of this paper is to apply this alternative approach.

The four RDBMS systems chosen for design review, Sybase, Oracle, Ingres and Informix, are leaders in the UNIX market. Non-relational systems and some other popular UNIX RDBMS's were excluded to keep the scope of the paper realistic. Five categories of design characteristics are considered. They are: 1) Query Processing - choosing the best method of finding the answer to a query using existing database structure, 2) Query Optimization - applying methods to improve the performance of query evaluation algorithms, 3) Distributed Query - the ability to tie together data that already exists but that resides on different machines, 4) Concurrency Control - how the system maintains the integrity of several transactions executing at the same time and, 5) Crash Recovery - in the event of a failure, how does the system guarantee its ability to recover? The core question to be addressed here is "Can one make a choice of which RDBMS to benchmark on the basis of a comparitive design review?"

Query Processing

Given a query, there is generally a variety of methods for computing the answer. While one would like users to express queries in a way which suggests the most efficient execution, this is often unrealistic. As a result, it becomes incumbent upon the system to transform the query as entered by the user into an equivalent query which can be computed more efficiently. Typically, the first action the database system must take is to translate the query from the language the user understands, either query language, 4GL or host language, to an internal form, such as relational algebra. The syntax of each query is checked, the relation names appearing in the query are verified, it is compiled, and a processing plan for the query is generated. Two of the techniques used to estimate query processing costs are statistics and indicies. Statistics can provide an estimate of the size of expression results, but updating them after every transaction is expensive. Indicies allow fast access to records but they also impose the additional overhead of access to the blocks containing the indicies. These techniques are used for query optimization, which is treated in the next section.

In the case of Sybase[1], query processing takes the following form. First, the parser checks for correct grammar and syntax. Object names are then validated against the information in the data dictionary: do the tables and columns in the query actually exist, and are the computations called for reasonable? Next the query is normalized and translated into a standard form, and

CH2728-4/90/0000/0441$01.00 © 1990 IEEE

protections are checked. Finally, the query is optimized and compiled. Sybase depends on the underlying operating system less than 10% of the time[3]. Its dataserver requires less than 34kB of memory for each additional user. Sybase studies have shown that short queries spend 40% of their execution times pre-compiling (mainly parsing), 40% compiling, and 20% executing. Larger queries spend proportionally more time executing. Sybase supports both statistics and indicies as well as several query processing options. These include: 1) PARSE-ONLY, which checks the syntax of each query and returns any error messages to the host, without generating a sequence tree, compiling, or executing the query, 2) SHOWPLAN, which generates a description of the processing plan for the query after it is compiled, and continues processing it (unless NOEXEC is set), 3) NOEXEC, which causes the DBMS to process each query through the compile step but not to execute it, and 4) STATISTICS, which displays performance statistics after each query. These options present the user with feedback which can be used to improve performance by refining the query or making changes in the database design. Sybase offers a B-tree indexing scheme only, but allows additional user-setable configuration parameters to improve performance.

By clustering data from multiple tables on the same database page, Oracle speeds disk and data access during the execution of queries. Multiple indicies are automatically maintained and optimized as part of this DBMS's access plan[18], which includes not only the relational operations to be performed but also the indicies to be used, the order in which the tuples are to be accessed, and the order in which operations are to be performed. Indicies impose overhead however, in the form of increased size and some amount of additional access time. Oracle and Sybase both support index clustering.

The choice of a disk access mechanism, hashing, B-tree, heap or ISAM, can impact the performance of a UNIX DBMS. Ingres offers all four disk access mechanisms. All data definition information is catalogued in an integrated data dictionary which can be accessed with SQL. Queries are, by definition, one of the following three types: 1) single table, 2) one row (in one table) to one row (in another), 3) one row (in one table) to many rows (in another). An application sends a query to the distributed data manager which optimizes the query and dispatches sub-queries to revelevant individual data managers. Ingres maintains statistics on all tables, and uses indicies which can be created on any field or combination of fields[15].

Informix emphasizes large scale, centralized database access by dozens or hundreds of users[23]. Query processing is described as consisting of validating users logic and querying the database.

Query Optimization

Response time and ease-of-use are normally viewed as being among the most important attributes of an information system. Query optimization tries to minimize response time for a given query language and mix of query types. It also can improve the ease-of-use of a DBMS by providing the user with feedback and other tools which can be used to add efficiency to the system. Communication costs, frequent disk accesses, and long computation times are the principal targets of query optimization and the tradeoffs between them can influence the strategy which is best for a particular application. In general though, query optimization techniques try to: 1) avoid duplication of effort, 2) use standardized parts, 3) look ahead in order to avoid unnecessary operations, 4) choose the cheapest way to execute elementary operations and, 5) sequence them in optimal fashion[5]. In distributed database systems, optimization must pay particular attention to the tradeoffs between communication and local processing costs, network architecture and data distribution strategy.

Sybase's query optimization techniques include "stored procedures", query processing options, configuration parameters, and a library of host language subroutines and procedure calls. Faster execution times generally result from the fact that SQL queries and transactions are compiled rather than interpreted. Users define stored procedures which are parsed, validated, normalized, and compiled with their execution plans the first time that they run. Subsequently, it is only necessary for the Sybase dataserver to locate the procedure, go through its protection checks, and substitute parameters before starting execution. These procedures can be invoked with a single remote procedure call in distributed applications, reducing performance penalties due to network traffic by up to 80%. Query processing options allow a database user to estimate the amount of time and resources required by a query or transaction. As mentioned, Sybase includes "parse only", "showplan", "noexec", and "statistics" options. Statistics available include: 1) Total time CPU and I/O are busy, 2) total time a dataserver is idle, 3) number of packets, bytes and errors received and sent, 4) number of read and write operations, 5) number of connections made, and 6) number of processes that currently hold locks. Configuration parameters are normal-

ly established by the database administrator. Among those offered by Sybase are clustered B-tree indicies for faster disk I/O, tunable procedure and buffer cache size, and control over the average time that any task will dominate the CPU.

Oracle speeds disk and data access by physically locating data from multiple tables on the same database page. When you add, delete, or change data, Oracle automatically maintains and optimizes indicies. Oracle's database administrators monitor utility provides statistics on: 1) lock status, 2) log file status, 3) logical and physical I/O activity, 4) users using the DBMS, 5) programs each user is running, and 6) tables being accessed.

Ingres applies statistics, indicies, intelligent buffer management and "repeat queries" to optimize query processing. The Ingres data dictionary stores all performance related information - table structure and size, available indicies, and data distribution statistics - and allows the database administrator (DBA) to change the physical characteristics to improve performance. The query optimizer has access to this information and a knowledge base of rules used for optimization of the query execution plan. Additionally, indicies can be created on any field or combination of fields. The query optimizer checks each index to make sure "it knows" the fastest way to access the desired data. An index in Ingres can be modified to any storage structure, independent of the base table's structure, to increase performance. Ingres' intelligent buffer management system also increases performance by keeping frequently used pages in memory on a per user basis to minimize the amount of disk I/O. The size of the buffer can also be tuned. In addition, the amount of data per page can be adjusted, multiple disks are supported, and a tunable lock manager sets the maximum number of users working on a database. Finally, "repeat queries" are cached for faster execution.

Informix query optimization focuses primarily on techniques which minimize physical disk I/O. These include shared memory, direct (raw) disk I/O, cluster indexing, and sorted writes. Data pages in Informix are read into shared memory and the records contained on a page are available to all transactions. Once a record is in shared memory, it may be modified or read by multiple users without being refreshed from disk[13]. Data is stored on raw devices, not the UNIX file system, allowing contiguous data pages. During a checkpoint (point at which pages of memory which contain modified database records are written to disk), pages are sorted first to maximize the number of pages written per I/O. Cluster indexing forces the physical order in a table to be the same as the order of an index, enhancing the speed of the indexed retrieval. In addition, tables and databases can span multiple disk drives allowing parallel access of frequently used tables. Like the other leading UNIX RBMS's, Informix also provides performance monitoring of: 1) system parameters, 2) performance, 3) tunable shared memory, 4) locking (controlled), 5) indexing (controlled, and 6) locations of tables (controlled.

Distributed Queries

By one account, " a single-site database management system will be an antique curiosity within the next ten years"[10]. Rather, distributed fragments of one logical database will reside on several physical databases providing the following benefits: 1) remote sites will share the same global view of the database but each site will concern itself only with maintaining local information, 2) redundancy of data, through duplication of local databases, will add robustness to the overall database, and 3) I/O, CPU and network resource demands will be distributed. But today, distributed database access is held to be "difficult to design properly and maintain"[11]. With a two phase commit, a distributed DBMS can successfully recover from all single and multiple-site failures and certain cases of network partitions. But the two-phase approach requires an extra round of messages in the protocol, thus there is a trade off between the level of service and the cost. When multiple copies of a distributed object (local databases) are updated within a transaction, a read can be directed to any convenient copy but a write must update (and lock) all copies. Not until the transaction commits can all the write locks be released. This can have serious performance implications if, for example, the distributed database involved maintains one local copy in New York and another in Hong Kong.

What is clear though in the case of Sybase is that the client-server distributed architecture is gaining increasing acceptance. In it, the user application code is separated from "back-end" database server functionality. Integrity checking and transaction logic are performed in the server, while application logic and the user interface occur in the client. Sybase claims the following advantages result: 1) performance, 2) integration, 3) economy, 4) scalability, and 5) reliability [22]. The degree to which these are in fact advantages for a particular application probably needs to be measured. Sybase's dataserver can also be shared by multiple dissimilar front-end computers and multiple databases can be open at

once. Sybase's implementation of distributed query allows access and update of data from multiple dataservers in the same transaction. At present, an application handles the two phase commit, although the plan is to incorporate this into the dataserver in a future release. Lastly, multistatement retrieval and update transactions accessing multiple databases on multiple dataservers are supported, with incomplete updates automatically rolled back[3].

Oracle supports remote application access with a "kernel residing only on nodes with a database." In this architecture, a single SQL references data at one location only[9], transactions are not coordinated across databases, and each application is responsible for table maintenance across nodes. As a result, although processing is performed on multiple machines, each logical database is located at one node only. Oracle does have support for parallel processors which allows for optimization of the Oracle background processes, reducing I/O bottlenecks and enhancing performance.

Ingres implements multiple database servers on any number of CPU's. The Ingres Star Distributed Data Manager receives an SQL and divides it into subqueries. Each local data manager executes its subquery and returns selected data to the application[16]. The following are also features of Ingres: 1) two phase commit, 2) automatic recovery, 3) table fragmentation, 4) distributed query optimization, and 5) parallel query execution. Ingres supports multiprocessor UNIX and most popular networking protocols.

Informix Turbo is built on a requestor/server model that separates the user interface (application) from the database server (engine). Each user has their own server process and thus multiprocessor hardware is supported (a single process server DBMS cannot distribute across multiple processors). "Most popular network protocols" on a "wide variety of heterogenous computer systems" are supported[12].

Concurrency Control

Interactive time-sharing systems which support a number of concurrent database transactions executing simultaneously must control the interaction among them to preserve the consistency of the database. A number of concurrency control schemes including locking protocols, timestamp ordering, validation, and multiversion techniques do so by either delaying an operation or aborting the transaction that issued the operation. A locking protocol is a set of rules which state when a transaction may lock and unlock each of the data items in the database. A timestamp-ordering scheme ensures concurrency control by selecting an ordering in advance between every pair of transactions. A validation scheme tests the validity of the unique fixed timestamp associated with each transaction and rolls back those which do not pass. Multiversion concurrency control also uses timestamps to ensure that a read operation always succeeds, while write operations may result in the rollback of a transaction.

Sybase implements user defined transaction control with BEGIN, COMMIT, ROLLBACK, and SAVE TRANSACTION SQL statements. Users also have the option of making normally shared locks more restrictive. Both deadlock (one transaction waits for resources held by another, which itself waits for resources held by the first, in an infinite loop) and livelock (an exclusive lock is prevented from aquiring resources because a series of shared locks keeps interfering) are automatically detected and handled by the dataserver[3]. Sybase handles integrity enforcement by allowing table owners to create "stored procedures" which ensure that whenever changes are made to tables by applications, none will violate the consistency required by foreign key values and the primary key values they reference.

Oracle's concurrency control mechanism is based on the use of shared memory to contain database locks, buffers, cache, and queues. Oracle does not depend on UNIX for locking, rather semaphores and signals arbitrate locks providing a more granular record locking scheme. (It is not clear that record locking is an advantage when transactions are distributed across the pages of a database.) Oracle's integrity control scheme uses the UNIX "write-through" buffer cache and, for those systems that do not support it, raw devices are used to write data to disk on demand.

The Ingres database lock manager implements read and write locks on the entire database, a single table, or a single data page. Ingres automatically escalates locks when necessary and provides automatic deadlock detection and rollback[15]. Locking rules can be adjusted. Ingres enforces referential integrity by "looking up entered values in database tables."

Informix applies locks to either the entire database, a table, a page, or a row. Read locks, which provide "degrees of isolation from other transactions", include dirty read, committed read, cursor stability, and repeatable read[12].

Crash Recovery

An integral part of a database system is a recovery scheme which is responsible for the detection of failures and the restoration of the database to a consistent state that existed prior to the occurence of the failure. Put another way, it is the responsibility of the recovery scheme to ensure that all of the instructions associated with any transaction are executed to completion, or none are performed. A common way of doing this is to log all transactions on disk or tape before the database is updated. In the event of a failure, the log can be used to restore consistency. This technique is also called journaling. A less common method of ensuring consistency is shadow paging, in which uncommitted transactions are mirrored on disk. Crash recovery in a distributed database is more difficult since each transaction has to be committed everywhere or aborted everywhere. The two phase commit strategy is viewed as a way of solving this problem, but it is not yet implemented in all of the leading RDBMS's and it can impose a longer response time.

Sybase's dataserver recovery features include a write ahead log and a user setable "maximum recovery time" from which an appropriate checkpoint interval for writing "dirty pages" to disk is derived. In the event of a failure, all transactions that were in progress but not yet committed at the time of the failure are undone. Completed transactions are redone if there is no guarantee that they were committed since the last checkpoint. Sybase provides a software implementation of disk mirroring for either the transaction log or the entire database, and a "dynamic dump" utility allows the database and transaction log to be backed up while in use, encouraging frequent backups.

Oracle stores an "after image" journal on a separate disk to provide crash recovery. Applying the journal, a "roll foward" recovery, results in the writing of committed transactions to a backup copy of the database. Uncommitted transactions are rolled back.

Ingres and Informix also implement checkpoints and journaling. Informix keeps both "before image" and "after image" logs to facilitate fast recovery of all complete transactions.

Conclusion

Can one make a choice of which DBMS to benchmark on the basis of a comparative design review? Perhaps. Based on the material considered here, Sybase appears to have the best distributed architecture, the most advanced transaction control, integrity checks, and crash recovery features. Are there additional issues to consider? Of course. Application development tools, security features, portability, cost and support have not been addressed. Also, in making this kind of decision, savvy users will attempt to find someone who has successfully built an application with the favored DBMS. It's fair to say though, that the database systems reviewed here are all enjoying increasing acceptance along with UNIX, workstations, and minicomputers in general. Any of them could offer significant advantages if carefully applied to a given problem. Understanding the design issues presented here represents the first step toward making an informed decision and going on to a meaningful benchmark.

References

[1] J. Bowman and K. Paulsell, "The Datatoolset," Sybase Technical Report 3010-3.0, March 1988.

[2] L. Boyd, "Standing on Shaky Ground? RDBMS pioneer Sybase trading for top share in distributed databases," The Sun Observer, September 1989.

[3] M. Darnovsky and S. Emerson, "The Dataserver," Sybase Technical Report 3000.3.0, March 1988.

[4] J.L. Hursch and C.J. Hursch, Working With Oracle, Tab Professional and Reference Books, 1987.

[5] M. Jarke and J. Koch, "Query Optimization in Database Systems," Computing Surveys, vol. 16, no. 2, June 1984.

[6] F. Pascal, "Sybase Today, SQL Server Tomorrow," Data Based Advisor, April 1988.

[7] U. Rodgers, "Benchmarking Techniques for UNIX DBMS's," Database Programming and Design," November 1988.

[8] S. Roti, "SQL + 4GL = Informix," Data Based Advisor, September 1988.

[9] L. Schaeffer, "Oracle for UNIX - General Information Manual Version 6.0," Oracle Technical Publication 449-V6.0, August 1988.

[10] M. Stonebraker, "The Distributed Database Decade," Datamation, Sept. 1989.

[11] "Distributed Databases," Sun Microsystems Software Technical Bulletin, August 1989, p. 1055.

[12] "Informix Turbo: OLTP Performance for the Real World," Informix Software publication 700-000-001-0, 1988.

[13] "Informix Turbo. On-Line Transaction Processing Database Engine Version 1.10.03," Informix Software publication 732-000-009-0, 1989.

[14] "Ingres/Applications," Relational Technology pub. MIV-002A-001, 1987.

[15] "Ingres Base Product," Relational Technology pub. MIV-001R-001, 1987.

[16] "Ingres - Overview," Relational Technology pub. MIN-0671-002, Dec. 1987.

[17] "Ingres Release 6.0," Relational Technology pub. MIV-010S-001, Dec. 1987.
[18] "Oracle: Products and Services Overview," Oracle pub. 19172.0189 1250, 1989.
[19] "Oracle: The Relational DBMS Solution for UNIX," Oracle pub. 19570-0288, 1988.
[20] "Performance Evaluation of an On-Line RDBMS," Sybase pub. 5012.0588, 1988.
[21] "SQL Server," Sybase pub. 3040.0189, 1989.
[22] "SQL Toolset," Sybase pub. 3030.0189, 1989.
[23] "UNIX On-Line Transaction Processing - A Brief Definition," Informix Software 1989.