Beyond OLTP: On-Line Complex Processing

Many companies need On-Line Transaction Processing (OLTP) and the DBMSs that can support it. Nonetheless, OLTP applications are only a special, albeit historically difficult, category of application to support with a relational DBMS. As the business community moves to respond to a market with needs that may change from minute to minute, companies will not be able to compete without what is called On-line Complex Processing, or OLCP. The database management systems which support OLCP applications will have to be based on the relational model or some extension to it. Over time, OLCP may well replace the role played at present by OLTP.

by David McGoveran

This article defines OLCP, discusses the requirements for an OLCP DBMS, and documents a set of rules that can be used to measure how well a DBMS meets the needs of OLCP environments. This list of objectives is not meant to be exhaustive, nor are they meant to provide a "scoreboard" for DBMS evaluations. They can, however, be used to guide the DBMS selection process and to provide a foundation on which to build further evaluation criteria.

As OLTP is still very popular, the reader might legitimately ask, "Why introduce yet another buzzword and another goal for the industry?" There are several reasons:

- The article "OLTP: What is it?" by Colin White, which appeared in the Spring 1989 issue of InfoDB presented objectives for a relational DBMS that supports an OLTP environment. In the process of evaluating DBMSs against these objectives, we have learned a great deal about the kinds of support available today. Support for OLTP by relational DBMSs is far from perfect, however. Part of the reason is that DBMS vendors are attempting to provide OLCP facilities in their products, and some of these conflict with OLTP requirements.
- A recent study by the Aberdeen Group in Boston found the performance needs of 90% of all OLTP applications could be met by a relational DBMS delivering the equivalent of 12 debit/credit transactions per second. If this is true, then why do businesses still find performance to be an essential criterion in selecting a DBMS? We feel that it is because of OLCP needs, which are not addressed by OLTP requirements.
- There is a need to characterize the databases and database transactions likely to be found in an OLCP environment. We do this not only to define OLCP requirements, but also to contrast these requirements with those for OLTP.
- Finally, we are convinced that the complexity of applications will continue to increase at a phenomenal rate over the coming decade:
  - Users are becoming more familiar with the benefits of graphical user interfaces, artificial intelligence for query processing, and text, graphic and video databases.
  - The inability to train end-users in a cost and time effective manner is promoting the use of smarter user interfaces and applications, which eliminate as much human involvement as possible.
  - Businesses are attempting to integrate more and more of their operations into a common logical database using data-driven applications.
  - The application backlog is promoting the development of re-usable software driven by a data dictionary.

An OLCP Example

As an example of OLCP requirements, consider a Wall Street on-line trading and portfolio management system. Brokerage houses make money on the volume of trades which they are able to "structure." In simple cases, trades are a matter of simply buying X and selling Y between two parties at an agreed price. But, in today's world, deals are often more complicated. When to sell or buy depends on a complex set of parameters, and sometimes on proprietary algorithms and heuristics, which the brokerage house follows.

We have seen traders monitor as many as twelve video display terminals at once, handling three or four telephone lines, and performing rapid computations in order to buy or sell as rapidly as possible. Automating this process requires on-line access to a great deal of numerical data, precise numerical computations, coded character field manipulations, and, most important, database schema flexibility.

The very nature of the brokerage business is change — new securities are invented all the time. Once the characteristics of a security are identified, it is necessary to perform all computations consistently. This demands a flexible database schema. If the entire business is to be truly automated, new securities have to be modeled on-line. There is no time for a data modeler to determine how to store the relevant information and renormalize the database. Today, most of this security information is stored in a passive manner, much like a text file would be stored. This barely meets requirements and is

only slightly better than recording the information on scraps of paper for batch data entry and overnight processing.

The brokerage business is highly competitive. Creating a new kind of security is like designing a new product: if it is more attractive than other products it sells, and gives its creator a competitive edge. Unlike manufacturing, securities are "paper" products. They can be created almost as fast as someone can think of them. This situation drives the need for flexibility. The volume of trade drives brokerage profitability, which in turn means massive amounts of data regarding trades and quotations need to be collected, queried, and analyzed

There are a number of other examples of OLCP applications, including:

- bill of materials explosion
- manufacturing-dynamic scheduling and routing
- risk management — portfolio analysis and portfolio optimization
- CAD/CAM
- elementary particle research
- insurance policy maintenance
- insurance claims adjustment and reconciliation
- telecommunications provisioning

What Is OLCP?

For all the complexity that OLCP transactions add to relational DBMS requirements, the high availability and performance requirements supported by OLTP are not lessened. In fact, they may be increased over that of OLTP environments. Furthermore, there may also be OLTP, ad-hoc query, and decision support applications within an OLCP environment, all using the same highly volatile data.

OLTP database environments can be characterized as including:

- a relatively stable database schema and data model
- performance needs measured in 5 to 1000 transactions per second
- simple database access: narrow columns and tables, few tables and columns accessed and updated
- record-at-a-time updates and queries
- a few large (i.e. deep) volatile tables
- thin transactions — few statements per transaction
- high availability
- sophisticated recovery and tracking
- batch reporting and optional batch updating
- relatively straightforward integrity constraints

OLCP database environments, on the other hand, add complexity by having:

- an unstable (time-dependent and unpredictable) database schema and data model, possibly requiring multiple versions
- complex database access: many tables affected; many columns updated, scanned and selected; many rows inserted, updated, deleted, scanned and selected; fat columns and wide tables; aggregate functions; numerical computations
- set-at-a-time processing
- fat transactions — many statements per transaction
- transactions of possibly long duration (e.g., days)
- many large volatile tables
- high availability
- on-line decision management (OLDM) requirement
- complex integrity constraints
- a hybrid environment: decision support, interative ad-hoc query, OLTP, batch operational processing, batch end-user processing

OLCP Objectives

As with White's OLTP rules, the OLCP rules are designed to complement existing rules for determining other DBMS requirements, such as support for the relational model, and distributed database capabilities (where appropriate). However, it should be noted that DBMS requirements for OLCP applications are less tolerant of certain deviations from the relational model — entity, domain, and referential integrity must be supported. Some means (such as stored procedures and triggers) for implementing or simulating logical independence beyond updatable views is required, since the schema underlying an OLCP application is subject to frequent change.

The objectives for OLCP as they relate to relational DBMSs are similar to those for OLTP. In fact, OLCP objectives include almost all the objectives of OLTP. The objectives in this article cover five fundamental aspects of OLCP:

- **Performance** — OLCP products should provide good on-line performance.
- **Continuous Operation and High Availability** — OLCP products should reduce or eliminate planned and unplanned outages.
- **Architecture** — OLCP products should have an overall architecture that does not impose limits upon the nature or volume of transactions to be processed and should provide capacity for growth.
- **Systems Management Controls** — OLCP products should provide facilities for controlling security, auditing and performance.
- **Processing Flexibility** — OLCP products should provide flexible and robust processing facilities.

The OLCP objectives defined in this article modify and extend the objectives defined in the original OLTP InfoDB article — see Figure 1 for a list of the original OLTP rules. Unless stated otherwise, the OLTP rules in Figure 1 apply also to the OLCP environment. OLCP rules which replace the OLTP rules or sub-rules are marked with an *R*; rules which extend the OLTP rules or sub-rules are marked with an *E*.

Performance

Rule 1. Efficient Concurrency Scheme

1.1R Concurrency scheme supports *user-controlled* locking granularity when *reading* data. Controls are by

- system
- application
- transaction
- statement

Lock granularity when reading data can be by

- record
- page
- table
- predicate
- index
- index page

Rule 1. Efficient Concurrency Scheme

1.1 Concurrency scheme supports record level locking when reading data.

1.2 Update transactions do not affect the performance of read-only transactions, and vice versa.

- multi-version read
- "dirty read"

1.3 Concurrency scheme supports record level locking when modifying data.

1.4 System handles and recovers from deadlocks.

- deadlock detection
- timeout

1.5 Excessive lock wait times are prevented.

- returns to application if a resource is locked
- maximum lock wait time system parameter

1.6 Transactions cannot degrade system throughput by taking too many locks.

- maximum locks system parameter
- lock escalation

1.7 System supports multiple levels of transaction consistency (as defined by ANSI/ISO SQL2 proposal).

- Level 0 (dirty read)
- Level 2 (CS)
- Level 4 (RR)

Rule 2. Efficient Commit Logic

2.1 No synchronous data page I/O during commit.

2.2 Supports group commit.

2.3 System coordinates the committing of database changes and data communication monitor messages.

- DB and DC in a single task
- DB and DC two-phase commit

Rule 3. Efficient Database Management

3.1 Supports look-aside buffering.

3.2 Supports (asynchronous) sequential pre-fetch.

3.3 Supports deferred write buffering.

3.4 Supports chained write buffering.

3.5 Supports parallel I/O operations.

- for write processing
- for read processing

Rule 4. Efficient Storage Management

4.1 Supports B-tree indexing.

4.2 Supports hashing.

- hashed index
- hashed data

4.3 Supports cross table combined index.

4.4 Supports intra-table data clustering.

4.5 Supports inter-table data clustering.

4.6 Supports data compaction.

- automatically compacts zeros and blanks in data
- supports index compression
- supports data compression exit routines

4.7 Supports on-line disk space reclamation and extension.

- space freed after delete operations
- allows dynamic extension of database size without impact on on-line application processing

Rule 5. Efficient Optimization

5.1 Optimizer uses a cost model and statistics.

5.2 Optimizer output is cached in memory for re-use by the same or different transaction.

- re-usable by the same transaction
- re-usable by a different transaction
- re-usable by a different application

5.3 Supports DBMS stored procedures.

5.4 Applications can execute DBMS stored procedures on a remote database server.

- client application can execute remote procedure
- procedure can call remote procedure
- called remote procedure is in the same transaction as the calling procedure

Rule 6. On-Line Utilities

6.1 Database utilities can be run while the OLTP system is active.

- load
- backup
- recovery
- reorganization

6.2 Availability of table data is not affected by utility operations.

- other tables remain available during table load
- other tables remain available during table backup
- table data can be read during backup operations
- table data can be updated during backup operations
- other tables remain available during table recovery
- other tables remain available during table reorganization
- table data can be read during index backup
- table data can be read during index recovery
- table data can be read during index reorganization

Rule 7. On-Line Definition

7.1 Definition can be done while the OLTP system is active.

- data definition
- security definition
- system definition

7.2 Availability of table data is not affected by definitional activity.

- other tables remain available while defining, altering or deleting a new table definition
- table data can be read while creating or deleting an index definition
- tables remain available when creating or deleting a view definition

Rule 8. Large Database Support

8.1 System allows a large table to be divided into subsets for utility operations.

- can load table subset
- can backup table subset
- can recover table subset
- can reorganize table subset
- allows parallel utility operations against subsets of the same table

8.2 System allows each table subset to be stored on a different device.

- subset is by file
- subset is by primary key
- subset is by index "key"
- subset is by SQL restriction

8.3 Supports restartable utilities.

- load
- backup
- recover
- reorganization

8.4 Supports parallel searching of disk volumes during query and update processing.

- single table read
- multi-table join
- index scan
- index update

Rule 9. Efficient System Recovery

9.1 System has efficient disk recovery logging (after image journaling).

- disk logging
- records physical data changed only

9.2 Full disk log automatically archived to archive volume without impact on system operation.

9.3 No manual intervention by the operator is required to restart system following system failure.

9.4 System restart times are controllable by the user.

- user can control system checkpoint frequency
- user specifies maximum restart time allowed

Rule 10. Efficient Database Recovery

10.1 Supports table level backup and recovery.

10.2 System tracks files required for recovery.

- backup
- logs
- backup information used during recovery

10.3 Supports table point-in-time recovery.

- table point-in-time-recovery
- ensures all related tables are recovered

Rule 11. Efficient Application Recovery

11.1 Backs out to last commit point after an application failure.

11.2 Supports application restart from last commit point after an application failure.

Rule 12. Fault-Tolerance

12.1 Supports duplexed logs.

- through hardware
- through DBMS software

12.2 Supports duplexed databases/ tables.

- through hardware
- through DBMS software
- duplexed table
- duplexed database
- duplexed disk

12.3 Supports "hot" standby processor.

12.4 DBMS supports sharing of database data between loosely coupled processors for availability purposes.

- DEC Cluster
- IBM
- UNIX
- other

Rule 13. Hardware/Operating System Software Exploitation

13.1 Supports multi-way processors.

13.2 Supports shared memory concepts.

Rule 14. No DBMS Imposed Limits

14.1 Unlimited database or table size.

14.2 Unlimited row size.

14.3 Unlimited number of table columns.

14.4 Unlimited number of table fields in an index.

14.5 Unlimited number of indexes on a table.

14.6 Unlimited number of connected or concurrent users.

14.7 Unlimited database bufferpool size.

Rule 15. Secure and Granular Authorization

15.1 Has field content level security.

- using views
- using security mechanism

15.2 Supports data access controls.

- read
- insert
- update
- delete

15.3 Allows grouping of users, resources and privileges.

- users
- resources
- privileges

15.4 Supports external security packages.

- user-ID and password
- database objects

Rule 16. Audit Capability

16.1 Records security violations by user-ID.

16.2 Records tables modified by user-ID.

- name of table modified
- actual data modifications

16.3 Records tables accessed by user-ID.

16.4 Records security definitions by user-ID.

16.5 Records data definitions by user-ID.

16.6 Records utility executions by user-ID.

16.7 Records audit data in easy to use form.

16.8 Tables can be selectively audited.

Rule 17. Performance Management Tools

17.1 Provides off-line performance reporting.

17.2 Provides on-line performance reporting.

17.3 Provides resource governor.

- CPU
- I/O
- disk storage
- memory
- records processed

17.4 Can control number of logged-on and concurrent users.

- logged-on users
- concurrent users

Figure 1. White's OLTP rules

Ideally, the DBMS could automatically determine the ideal lock granularity in any environment. In an OLCP environment, the mix of transactions and kinds of processing is so complex that it is unlikely an automatic system could determine the proper granularity dynamically. By supplying the user with sufficient control, concurrency can be improved for critical transactions.

1.3R Concurrency scheme supports *user-controlled* lock granularity when *updating* data. Controls are by

- system
- application
- transaction
- statement

Lock granularity when updating data can be by

- record
- page
- table
- predicate
- index
- index page

1.4E System handles and recovers from deadlocks:

- deadlock prevention
- automatic retry

The prevention of deadlocks is sometimes required. If this facility is not available, it is desirable to provide a mechanism for automatic retry of transactions which are aborted during deadlock recovery.

1.5E Excessive lock wait times are prevented:

- transactions can be prioritized

In an OLCP application, some transactions are more important than others. By assigning transactions a relative priority, resource waits can be controlled.

1.6E Transactions cannot degrade system throughput by taking too many locks. Lock escalation criteria are user-controlled by

- system
- application
- transaction
- statement

If a system provides lock escalation (or demotion) as a means of preventing transactions from taking too many locks, the criteria for lock escalation must be controlled. If the criteria are "hard-wired" into the system, lock escalation (or demotion) is more likely to reduce concurrency on critical data.

1.8E The user may influence the concurrency control mechanism:

- optimistic concurrency supported
- pessimistic concurrency supported

Most of the concurrency rules in the OLTP section assume a pessimistic concurrency control mechanism (i.e., using locking). In some OLCP applications, collisions between concurrent users are known to be unlikely and an optimistic mechanism may be more appropriate.

Rule 4. Efficient Storage Management

4.7E Supports on-line disk space reclamation and extension:

- dynamic disk space extension and reclamation can be disabled during critical operations

During batch inserts and deletes, the overhead of dynamic disk space management may be too great to allow completion of the batch work during off-peak hours.

4.8E The system warns the user when user-defined thresholds are reached regarding available and allocated disk space.

Rule 5. Efficient Optimization

5.1E Optimizer uses a cost model and statistics:

- uses cost functions (not cost indexes or heuristics)
- is not sensitive to statement syntax

The optimizer in OLCP applications must be sophisticated and should use true cost functions in optimizing database access. Because of the wide variety of Data Manipulation Language (DML) statements in the application, and their complexity, performance should not be sensitive to changes in statement syntax.

5.3E Supports DBMS stored procedures:

- accept parameters
- support error handling
- may be nested
- support a procedural language

Stored procedures provide a means of simulating logical data independence or implementing database schema independence for applications. This is important for schemas which are likely to change.

5.5E The DBMS supports both compiled and interpretive data DML modes of operation.

OLCP applications can consist of a combination of static and dynamic DML statements. The environment may contain OLTP as well as ad-hoc interactive applications. It is advantageous for the static DML to be compiled, and for the dynamic statements to be interpreted.

5.6E There is an EXPLAIN facility.

The ideal optimizer would automatically select the best access path regardless of the complexity of the DML, the database, or the application environment. Without such an optimizer, an EXPLAIN facility can be used to diagnose an optimizer error. The problem can then be corrected by modifying the DML or optimizer access path (see subrule 5.7).

5.7E Users may control the selection of the access path.

Direct user control of the access path selected can be used to correct optimizer errors and improve performance.

5.8E The method and frequency (e.g., sampling, continuous, periodic, or on demand) for gathering optimizer statistics is DBA controlled.

The updating of statistics used by the optimizer can be a costly operation. Some systems perform continuous updates, others only when a command is issued, and some allow for sampling of the table data. Depending on the size of the database and the temporal distribution of critical transactions, one update method may be more desirable than another.

5.9E Aggregate functions are optimized.

DML statements in OLCP applications contain a higher percentage of aggregate functions than in OLTP applications. The optimizer should be able to process aggregate functions efficiently and without redundancy.

5.10E The optimizer can optimize all active statements known to it:

- within a transaction
- across transactions

DML statements in an OLCP application may access the same data multiple times. The optimizer should be able to

detect this condition, cache the data, and prevent premature swapping to disk of the required data.

5.11E Indexes may be automatically created:

- supports temporary indexes
- supports permanent indexes

The automatic creation of indexes on read-intensive data can improve performance. Of the few systems which do this, most create temporary indexes which are dropped at the end of the DML statement or transaction.

5.12E The optimizer can distinguish between and optimize appropriately for various data value distributions.

Data value distributions are less likely to be uniform in an OLCP application than in other applications. Unless data value distribution statistics are maintained, the optimizer may incorrectly estimate the number of disk I/Os required.

Continuous Operation and High Availability

Rule 10. Efficient Database Recovery

10.4E Supports DML and DDL journaling and recovery.

Journaling Data Manipulation Language (DML) and Data Definition Language (DDL) statements can be more space efficient than journaling data when set processing is common. Also, if an error is made, the journal can be edited and reapplied to a backup. This is extremely important following a media failure in a high-volume update environment with large numbers of on-line data entry users. In such cases, re-running applications may not be feasible.

10.5E Supports transaction monitor supplied transaction numbers:

- journals the transaction number
- provides rollback to a specific transaction number

Database consistency does not require that the sequence of states of a database reflect the sequence of states of the applications environment at any point-in-time. For this reason, some means of identifying the order in which transactions are submitted by applications is needed. A transaction monitor usually supplies unique transaction numbers for each transaction submitted to the DBMS. It should be possible to rollback the database to a specific transaction in this order. In certain on-line update intensive and highly concurrent environments this may be the only way to manage the re-entry of data.

Architecture

Rule 14. No DBMS Imposed Limits

14.8E Unlimited (practical) number of allowed joins.

14.9E Unlimited (practical) number of tables referenced in a statement or transaction.

14.10E Unlimited (practical) number of expressions in a statement.

14.11E Unlimited (practical) number of subqueries in a statement.

14.12E Unlimited (practical) subquery nesting levels in a statement.

14.13E Unlimited (practical) number of cursors in an application or in the system.

14.14E Unlimited (practical) number of characters in a statement.

14.15E Unlimited (practical) number of columns referenced in a statement.

The DBMS should not impose unreasonable limitations on the complexity of DML statements or on the complexity of an application.

System Management Controls

Rule 15. Secure and Granular Authorization

15.4E Network transmission between clients and servers can be encrypted.

OLCP applications often involve data which must be kept secure. If the DBMS uses a client/server architecture with a network or other communications system interface, it should not be possible to tap into the communications link and intercept or simulate client-to-server or server-to-client communications.

Processing Flexibility

Rule 18. Data Definition Language Support

18.1E Temporary tables can be created which are automatically dropped at the end of the transaction or session.

The ability to create temporary tables as "working data storage" is especially important in complex transactions and in ad-hoc and decision support applications. It eliminates unnecessary application coding and communication between database and application.

18.2E Database triggers are supported.

Business or application rules that must be asserted against the database should not have to be coded in each application. Database triggers provide a means of asserting these rules and maintaining them within the database. Ideally, it should be possible to specify the event which causes a trigger to *fire*, not only on any DML statement, but also on a DDL statement and on utilities.

18.3E BLOBs, text, and image data types are supported.

OLCP applications frequently require storage and retrieval of non-standard data types. Especially useful are Binary Large Objects (BLOBs), text, and image data types.

18.4E User-created or abstract data types are supported with domain rules.

Support for user-defined (abstract) data types is especially useful in an object-oriented programming environment.

18.5E Timestamps are supported.

A timestamp data type provides a means of identifying the time and date of the last update of a row. This provides a means of tracking update events without explicit application code having to be written. It should not be possible to update the timestamp column directly using DML statements.

18.6E DDL statements can be used within a transaction boundary and can be committed or rolled back.

A database schema is subject to frequent change in an OLCP environment and costly mistakes can be made. Schema changes should, therefore, be managed as transactions.

18.7E Tables can be assigned a time of creation, a date when valid and an expiration date.

It should be possible to create a new schema which becomes the effective schema at a given date and time, and becomes expired at a defined date and time.

18.8E Tables definitions can be automatically assigned version numbers, and a certain number of inactive versions can be maintained in the database.

The tables modified in a schema change should have version numbers

so that changes can be audited effectively.

Rule 19. Data Manipulation Language Support

19.1E Data type conversion functions can be used in a DML SELECT list.

When schema changes result in the changing of a column data type, it should be possible to use data type conversion functions in the SELECT list. Fewer changes then need be made to the application code.

19.2E Potentially dangerous interactive statements which meet an installation-defined pattern are detected and require confirmation before execution.

In an OLCP environment which mixes ad-hoc users with high-volume critical applications, it may be necessary to highlight mistakes made by ad hoc users without actually denying the use of any given DML or DDL statement. By requiring confirmation from a user before executing certain statement types, unintentional errors can be controlled.

19.3E Error messages are sufficiently detailed to allow the end-user to debug ad hoc DML statements.

Interactive users writing complex DML statements need considerable help in debugging errors. It is not sufficient to produce a simple "syntax error" message. Identification of the location of the error and on-line help facilities describing correct syntax for the statement are required.

19.4E Informational messages such as the number of data rows affected by a statement, the processing time, and the type of statement executed are provided.

In OLCP applications, the conditional execution of successive statements may depend on the number of data rows affected by the previous statement. The system should provide a means for obtaining this information, as well as the type of statement executed, and the time required to process the statement.

19.5E Error messages can be logged along with user-id, application-id, statement, and a timestamp.

In high-volume environments with large numbers of on-line users, some means of analyzing errors by type, user, application, date, and time is necessary to make the system more robust.

19.6E A means is provided to increment a column value automatically during a set insert or update operation.

OLCP applications often manipulate data which has an intrinsic order. Such data cannot be inserted or updated using set operations unless an automatic and internal mechanism is provided for generating sequence numbers.

19.7E The results of a particular query may be processed by successive queries without explicitly creating a temporary table.

The ability to successively filter data for subsequent operations is a common requirement in decision support or "browse and update" applications. The application should not be burdened with having to explicitly create temporary tables to hold intermediate results.

Rule 20. Batch Support

20.1E Failed, idle, completed, or aborted batch processes are automatically detected and recovered:

- user definable timeout value
- error conditions detected

Batch as well as other processes may terminate in a variety of ways. The system should anticipate ill-behaved termination with respect to the database, detect it, and recover gracefully. It should not be possible for system resources to be consumed by such processes.

20.2E Bufferpool sizes may be dynamically configured to optimize input, intermediate, or output processing.

Tuning the allocation of bufferpool resources is especially important in batch operations. For example, an operation known to perform a large sort with little input or output would complete more quickly if most of the bufferpool were allocated to intermediate processing.

20.3E Batch insert (page append) is supported.

Certain applications require the insertion of large amounts of data in batch. This data should be accessible in the database in one contiguous piece. Furthermore, concurrency requirements may preclude the locking of significant portions of the target tables involved. One way to get around the problem is to support loading of the data into a temporary table. All pages of the temporary table are then allocated to the target table in a single operation and the the temporary table dropped. This process is called "page append."

20.4E A user determined batch size (number of rows affected) can be set to trigger an automatic checkpoint.

It should not be necessary to restart batch processes from the beginning in the event of a system failure. Issuing periodic checkpoints and logging progress can help.

20.5E Both DML and DDL statements can be included in batch processing.

Batch processing requirements can be as complicated as those of any on-line application. This is especially true when some time "window" exists for processing in batch without interference to on-line applications. Systems should, therefore, support the batch submission of arbitrary DML and DDL statements.

The Potential for OLCP

The combination of world market pressures, growth of cottage industries and tele-commuting, increased dependence on information, and rapid technological advances, all conspire to demand OLCP applications. At the same time, OLCP database applications offer a unique potential for today's businesses by closing the information loop between the customer and the factory. Businesses can, therefore, become more responsive to the market, and the time from application design to product introduction can be reduced.

No single DBMS meets all the needs of OLCP today. In fact, it is likely that OLCP performance requirements will exceed DBMS capabilities for some time to come. One thing is certain — only relational databases can meet the flexibility requirements of OLCP.

References

1. C.J. White. "What is OLTP?" *InfoDB*. Spring, 1989.
2. D. McGoveran. "The Power of Stored Procedures." *Database Programming and Design*. September, 1989.
3. D. McGoveran. "Evaluating Optimizers." *Database Programming and Design*. January, 1990.