DB2 Common Server: Technology, Progress, & Directions

Bruce Lindsay
IBM Almaden Research Center
650 Harry Road, San Jose, California 9512-6099
e-mail: bruce@almaden.ibm.com

Abstract - **Advances in hardware capacity and application requirements for increasingly sophisticated analysis and exploitation of information resources continue to create new challenges for database management systems. DB2 Common server, Release 2, addresses the evolving hardware environment and application requirements with facilities and features that support high performance database exploitation and extend the power of the Relational Model.**

While there are many challenges presented by the needs of database users to more effectively exploit their information resources, we shall concentrate on only a few of the technical accomplishments and directions of DB2 Common Server, Release 2. The topics we discuss include: the new storage architecture designed to address the growing performance gap between the CPU and the disks; the introduction of User Defined types and functions; extensions to SQL that enhance the user's ability to extract and manipulate information in the database; and the implementation of data integrity facilities that enhance data value.

The CPU <-> I/O Gap: *an emerging problem*

The rapid increases in CPU power, coupled with the availability of SMP platforms, has outpaced the performance increases of disk systems. In order to balance CPU capacity with disk performance, parallel I/O streams are needed to keep the CPU busy. DB2 Common Server has introduced a new storage architecture which allows tables, indexes, and Large OBjects to be striped across multiple disks. Large block and parallel read-ahead is used to match CPU processing to the disk latencies. Future development will exploit DBMS knowledge of data placement to support parallel query execution driven by parallel I/O streams from different disks.

Proceedings of the 21st VLDB Conference
Zurich, Switzerland, 1995

Objects Meet Database: *persistence isn't just disks*

The importance of type-based data behavior for developing applications that model the application domain must be carefully integrated into the realm of persistent data management. Many of the object-oriented techniques suitable for (non-persistent) programming language environments are not appropriate in the domain of databases. To begin the evolution towards object-oriented, hierarchial type systems, DB2 Common Server introduces facilities to define simple DISTINCT types which are represented by values of base types. In addition, strongly-typed user-defined functions over base and DISTINCT types allow users to extend the behavior of base types and define the behavior of DISTINCT types. As we evolve towards more comprehensive support for abstract data types, considerable care is needed to preserve the ability of applications to work with the data elements that they supply and extract from the database. A continuing challenge is to provide common type-based behavior for values in SQL queries and for values in the application program.

SQL Expressiveness: *extracting information value*

To enhance the ability of users to extract the maximum value from their stored data, DB2 Common Server has extended the SQL language to support several useful constructs. New support for Large OBject (LOB) base types allows users to store up to 2 giga-bytes of data in a single table field. LOBs are implemented using special storage and recovery mechanisms that (optionally) avoid logging of LOB values and allow the system to defer access to LOB data until the data is needed by the application.

Support for the ANS SQL2 CASE expression has been implemented in DB2 Common Server. CASE expressions allow users to code conditional expressions within a query and can be used to remap values and give data dependent results. We have also extended SQL syntax to allow *table expressions* within the FROM clause or at the head of a query. These view-like table expressions allow queries that would otherwise require the definition of a permanent view. Table expressions are also

used in the definition of *recursive* queries. DB2 Common Server support for recursive queries enables database users to compute transitive closures (and other recursive expressions) without repeated interactions between the application and the database.

Data Integrity: *the devil is in the details*

DB2 Common Server delivers support for check constraints, triggers, and extends the support of referential constraints. These facilities allow users to specify declarative constraints that are automatically incorporated into queries that modify the database. Check constraints insure that field values satisfy a specified predicate. Referential constraints enforce parent / child relationships and are supported (correctly) for cycles of parent / child tables. Triggers respond to modifications and have the power (and dangers) of production rule systems.

DB2 Common Server (unlike some systems) enforces the relational set semantics for updates leading to check or referential constraint processing. Set semantics require that the final state of the database must be independent of the order in which a set of tuples are updated. Whenever a table can affect what gets changed and is also affected by changes, special care must be taken to insure consistent and deterministic results.

Conclusions: *lots is done, more to come*

Release 2 of DB2 Common Server delivers exciting new function to relational database users. In addition, advanced optimization and execution techniques make it a leader in raw performance as well as in cost per transaction. Much of the new function in Release 2 is derived from technology developed in the Starburst project at the IBM Almaden Research Center. The technology of the Starburst system has enabled us to push back the internal complexity barriers that plague many full function database systems. For this reason, one can expect rapid evolution of function and additional performance enhancements in future releases of DB2 Common Server.