IBM's Relational DBMS Products: Features and Technologies

C. MOHAN

Data Base Technology Institute, IBM Almaden Research Center, San Jose, CA 95120, USA
mohan@almaden.ibm.com

Abstract This paper very briefly summarizes the features and technologies implemented in the IBM relational DBMS products. The topics covered include record and index management, concurrency control and recovery methods, commit protocols, query optimization and execution techniques, high availability and support for parallelism and distributed data. Some indications of likely future product directions are also given.

1. Introduction

Currently, there are four IBM™ relational data base management system (RDBMS) products. They are SQL/DS, DB2,™ IBM DATABASE 2 OS/2 -V1 and AS/400™ DBMS. While the SQL language supported by these systems is more or less the same [IBM92], the architectural and implementation features of these systems are very different. In this paper, we focus on some of the features and technologies of these DBMSs. We do this by considering the different aspects of data base management - namely, buffer management, index management, record management, recovery, concurrency control, query optimization, etc. More details about these can be found in the referenced papers. First, we briefly introduce each of the systems.

1.1. SQL/DS

SQL/DS, IBM's first RDBMS, was introduced in 1981. That release was essentially the product version of System R [ChGY81]. Since then, SQL/DS has evolved considerably. It runs on VM and VSE. In VSE, it can be accessed via CICS, with atomic update of CICS and SQL data guaranteed via two-phase commit (CICS acts as the coordinator).

1.2. DB2

DB2, first released in 1983, initially used the SQL/DS code for the upper parts of the system (e.g., query optimization and query processing features [CLSW84]). The DB2 data manager, on the other hand, was designed and built from scratch. DB2 runs on the MVS operating system and can be accessed through the transaction monitors IMS/TM and CICS. Atomic commit of updates on both DB2 data and CICS/IMS data within a single transaction is guaranteed. DB2 is tightly integrated with the features of its underlying MVS and hardware. DB2 V2R3 provides some basic support for the shared disks environment: multiple DB2s can read the shared data base or one DB2 can update it. Support for finer granularity of sharing [MoNa91] is being implemented.

SIGMOD /5/93/Washington, DC,USA

1.3. AS/400 DBMS

AS/400 DBMS, like its predecessor S/38 DBMS, is different from the other IBM RDBMSs in that most of its functionality is implemented in the hardware and in the licensed internal code of the machine [ClCo89]. This approach allows tight integration of system management functions and increases ease of use. Further, it permits the same data to be accessed concurrently through either the file system or the SQL interface, both for read and write [AnCo88]. SQL/400,™ which was first released in 1988, runs on top of the OS/400™ operating system.

1.4. DB2/2

The IBM Database 2 OS/2 -V1 (DB2/2 for short) runs on OS/2 [ChMy88]. It is the newest of the IBM RDBMSs. It was introduced in 1988 as OS/2 Extended Edition Database Manager. Its most recent 32-bit version and enhancements is DB2/2 V1. Most recently, the newest member of the DB2 family for the AIX/6000,™ called DB2/6000,™ has been announced. DB2/6000 is a port of the DB2/2 code base with substantial enhancements in function and performance. Since they share a common code base, these products will continue to evolve together.

2. Storage Management

DB2 and SQL/DS allow more than one table's records to be stored in the same file (and even on the same page). In those systems, when a record is inserted, a clustering index, if there is one, is used to determine the ideal page for inserting the record. An attempt is made to insert the record on that or a nearby page. DB2, DB2/2 and SQL/DS have special pages called the space map pages (SMP) or file space inventory pages (FSIP) which describe approximately the space available in a corresponding set of data pages. This compactly represented information is made use of while looking for a page with a certain amount of free space and while performing table scans to skip reading empty pages. In the case of DB2, it is also used to perform mass deletion of all records of a table efficiently [CrHT90]. AS/400 provides options to use record insertion into "holes" left by deleted records or to insert new records at the end of the data file.

DB2 allows a table to be partitioned based on key ranges of a partitioning key. For each partition, a local index is maintained on the partitioning key. All other indexes on the table are maintained as global indexes [ChMo93]. The partitioning index is also treated as the clustering index. SQL/DS and DB2/2 store all the indexes of a table in the same file. While the other DBMSs implement B^+-trees, AS/400 implements radix binary trees with front compression and, optionally, sparse indexing. AS/400 im-

---

™ AIX, AS/400, DB2, DB2/2, DB2/6000, IBM, OS/2, OS/400 and SQL/400 are trademarks of the International Business Machines Corp. X/Open is a trademark of X/Open Company Ltd.

plements a sophisticated disk striping scheme for storage allocation to load balance disk arms. To reduce the need for disk mirroring, it also implements a RAID-style checksum technique [ClCo89]. DB2 provides user exits which can be used to implement compression and validation of user operations on a per table basis.

By using record identifiers which do not refer to locations of records within a page, DB2, DB2/2 and SQL/DS provide flexible storage management. When garbage collection is done, records can be moved around within a page without the moved records having to be locked or the movements having to be logged.

3. Concurrency Control

All the DBMSs use locking, but the supported granularities of locking differ. All of them, except DB2, support record locking. DB2's finest granularity is a page for the data and a subpage (as small as .25K) for an index. DB2/2 uses ARIES/IM [MoLe92] for its index locking, while SQL/DS uses ARIES/KVL [Moha90a]. All, except AS/400, support automatic lock escalation and deadlock detection. AS/400 uses timeouts to handle deadlocks. To reduce the overhead of locking, DB2 V3 implements the Commit_LSN idea of [Moha90b].

In the area of isolation levels, all except AS/400 support repeatable read (RR). AS/400 supports an isolation level called read stability which is like RR except that new records are not prevented from being inserted by other transactions after a reader has locked records which satisfy a set of predicates (i.e., the *phantoms* are permitted). All support cursor stability (CS). DB2/2 and AS/400 support unlocked/uncommitted (*dirty*) read (UR) also. SQL/DS supports the finest granularity of specification for the required isolation level. It allows different statements of a transaction to be associated with different isolation levels.

When record locking is in effect, SQL/DS uses page locking for physical synchronization while DB2/2 uses page latches. AS/400 uses row level latches. From V3, DB2 does page latching even though record locking is not yet implemented. This permits the implementation of the Commit_LSN optimization and unlocked reads by the archive and statistics gathering utilities.

AS/400, DB2 and DB2/2 permit a cursor's position to be retained across the commit of a transaction.

4. Recovery

SQL/DS uses the shadow-page technique [GMBLL81] for recovery, while the others use mostly write-ahead logging. DB2/2 uses ARIES [MHLPS92], while DB2 originally implemented [Crus84] *selective redo* then implemented ARIES. DB2/2 uses shadowing for handling updates to long fields [LeLi89]. While AS/400 does write-ahead logging, it does not store log sequence numbers (LSNs) on pages. Hence, it does physical logging of updates to data pages. In the case of indexes, AS/400 uses shadowing using a physical log [DHLPR89]. When an index page is updated for the first time after an index's last checkpoint, its before image is logged. On restart after a failure, the physical log is used to reestablish an index's last checkpoint state. Then, any required index redos and undos are performed logically by using the log records written for data page changes [MoLe92]. If the system is used with index logging turned off, then, after a system failure, index recovery is performed by rebuilding the index.

DB2 supports archiving of data at the granularity of a partition of a table, while AS/400 supports table granularity. AS/400 users can also choose to save the log and thus get only the delta saved. DB2/2 and SQL/DS force the granularity to be the whole data base. While AS/400, DB2 and SQL/DS allow archiving to go on concurrently with updates, DB2/2 quiesces all accesses to the data base during an archive. SQL/DS archives the shadow version of the data base. Hence, a checkpoint is not allowed to occur while an archive is in progress. This is because a checkpoint discards the shadow data base and makes the current version of the data base become the shadow also. SQL/DS and DB2 (V3) might archive uncommitted data. DB2 permits page-level incremental archiving (i.e., archive only the pages modified since the last archive) [MoNa93]. DB2 is capable of recovering online damaged pages of a table from an archive while other pages of the table are being modified by transactions.

5. Buffer Management

Except for AS/400 DBMS, the other DBMSs have a traditional buffer manager. AS/400 DBMS relies on the system's single-level storage model. This means that the paging subsystem is the buffer manager and the LRU page replacement policy is applied across DB and nonDB pages in real memory. Hence, AS/400 DBMS does not perform DB-page I/Os explicitly. The paging subsystem supports sequential prefetch, but is unaware of recovery requirements. It does not call the DBMS logic before writing a dirty page to its home location on disk. As a result, to enforce the write-ahead log protocol, the DBMS leaves a modified page pinned in main memory until all its log records are written to disk.

DB2/2 uses the OS/2 file system, while DB2 uses a low-level (media manager) interface of VSAM. SQL/DS manages space within VM minidisks. It also supports the exploitation of *expanded storage* (i.e., page addressable storage which can be greater than the 2GB limit imposed by 31-bit addressing for the byte addressable main storage).

From the beginning, DB2 has had a very sophisticated buffer manager [TeGu84]. It supports multiple buffer pools, expanded storage, background batched writes of *dirty* pages, and sequential and skip-sequential prefetching of pages using system agents. It uses 2 LRUs to treat differently those pages which are read in using random I/Os and those that are prefetched.

SQL/DS forces to disk all dirty pages at checkpoint time. The others take fuzzy checkpoints.

6. Referential Integrity

DB2, DB2/2 and SQL/DS support the automatic enforcement of referential integrity constraints. Referential constraint violations are checked as each record is updated, deleted or inserted. In DB2, constraint enforcement is the responsibility of the data manager [CEHH90]. An operation is performed and then a check is made to see if there is a violation. This approach makes possible certain locking optimizations [Moha90b]. During the execution of DB2's load utility, the user has the option of requesting the suspension of constraint enforcement. If that option

is exercised, then a check utility must be run before the table can be used. The latter performs its checking more efficiently using a set-oriented approach.

SQL/DS and DB2/2 enforce constraints at the query compiler level. It is done by modifying a user's update statements at compile time by adding some query blocks that execute at query run time to check for constraint violations. When a uniqueness constraint is defined for a key, SQL/DS and DB2 obey the ANSI standard and defer checking for violations until the end of execution of an update statement.

7. Query Optimization

All the DBMSs support precompilation of SQL queries and perform cost-based query optimization. DB2's optimizer uses a very detailed and sophisticated cost model for estimating query execution costs [Moha92]. Like System R, DB2 and SQL/DS perform exhaustive search of the possible execution plans for a query, using a dynamic programming algorithm. DB2/2 and AS/400 use greedy algorithms. DB2 supports the tracking of frequently occurring values of indexed fields to avoid/reduce the errors in estimation of intermediate table sizes otherwise resulting from an assumption of uniform distribution of field values. SQL/DS takes a different approach and tracks boundaries of ranges of key values such that all the ranges have the same number of instances. DB2 considers the use of multiple indexes of a single table both for index anding and/or oring [MHWC90]. DB2/2 considers only index oring.

AS/400 and DB2 may choose a different plan if they are informed that the user is interested in only a certain number of records in the query result. AS/400 considers building temporary indexes to process a query efficiently. It may also stop looking for better plans if time spent in optimization exceeds a certain threshold. Unlike the others which use only statistics gathered during the previous execution of a utility, at the time of optimization AS/400 looks up indexes of a table to estimate the number of qualifying records [ACDL88]. It may also reoptimize a valid compiled plan at run time if it suspects that due to some recent changes to the data, reoptimization might be beneficial. The nested loop and merge scan join methods are supported by DB2, DB2/2 and SQL/DS. DB2 also supports the hybrid join method [CHHIM91]. AS/400 supports the nested loop join method in a lower level of its data manager. DB2/2, unlike the others, considers as the inner table of a join even composite tables (i.e., a result of one or more joins).

8. Query Execution

During execution, based on certain thresholds, DB2 might decide not to use some indexes which the optimizer had included in the plan [MHWC90]. By using index anding/oring techniques and the hybrid join method, DB2 converts many random, single-page I/Os into sequential, multi-page I/Os. Even when the DB2 index manager is not asked to do range scans, it caches some information about the most recent index access made by a transaction to see if the next access to the same index can avoid traversing the index from root to leaf. Unlike System R, DB2's data manager supports certain set-oriented operations (e.g., delete all records that satisfy a certain predicate) to minimize the number of inter-component interface crossings. DB2 and AS/400 are able to attain better performance by executing the DBMS code in the user process itself. This can be done without *wild* user applications causing integrity problems due to some features available in MVS and OS/400.

For sorting, DB2 implements tournament, quick and tag sorts. AS/400 supports tournament sort, while DB2/2 supports height-balanced and tag sorts, and SQL/DS supports quick sort. For queries that involve aggregation operations, DB2 improves performance by computing some aggregates during sorting.

9. Utilities

Utilities are used to perform batch operations like load, unload, reorg, index build, check consistency of index and record data, archive, recover, collect statistics, etc. Instead of going through the normal data manager interface which SQL accesses go through, the DB2 utilities interact directly with the buffer manager, thereby vastly improving performance. In DB2 V3, for partitioned tables, partition independence is provided so that utilities can be executed concurrently against different partitions of a table. AS/400 also supports index builds concurrently with table updates. These approaches improve availability and allow parallelism to reduce response times and to exploit MPs. Partition level, rather than table level, locking is used in DB2. Global indexes on such a table require special handling by the utilities to make such a level of concurrency possible.

10. Distributed Data

Distributed relational data architecture (DRDA) protocols [IBM90] are used when a client application executing in one environment (e.g., DB2/2) needs access to data stored in another (server) environment (e.g., DB2). With the initial DRDA protocols, within one transaction, data resident in an instance of a remote DBMS can be accessed. Such an access is called *remote unit of work*. All the products support the requester portion of the DRDA protocol and DB2, SQL/DS and AS/400 can be DRDA servers. DB2/2, in addition to being a DRDA application requester, is a server to lan attached clients.

Amongst DB2 systems, the more flexible *distributed unit of work* is also supported. In a single transaction, DB2 V2 supports access to data distributed across multiple DB2s, as long as each SQL statement refers to data stored in a single DB2. Since a single phase commit is used, only one site can be updated in a single transaction. But in V3, multiple sites can be updated since a generalized version of the *presumed abort* (PA) commit protocol [MBCS92, MoLO86] has been implemented. This protocol is now part of the SNA LU6.2 architecture. PA is also featured in the OSI and X/OPEN™ distributed transaction processing standards.

11. Parallelism

All the DBMSs allow multiprocessors (MPs) to be exploited. In the case of SQL/DS, while application processing can exploit MPs, executions of the data base code can only exploit a single processor since all data base accesses for a particular data base are performed by a single virtual machine (the SQL/DS server machine). None of the DBMSs provides intra-transaction parallelism, except that

DB2 V3 supports I/O parallelism. I/O parallelism allows a single user process to initiate I/Os to multiple partitions of a table in parallel and process the pages as they come in. This way, the speed mismatch between the CPUs and I/O devices is balanced.

12. Futures

In the Starburst research project [LLPS91] we have learned a lot about high performance complex query processing, enhanced concurrency and recovery techniques, and object oriented relational extensions. As we have done in the past (eg. exploiting ARIES technology in our DBMS products), we will continue to incorporate portions of the Starburst technology and code in further enhancements of IBM database products.

Also, SQL can be expected to be enhanced to support non-traditional applications more easily and flexibly. The Starburst project has provided us experience with respect to SQL enhancements needed by such applications. We would also expect to see enhancements in the area of connectivity to IBM and non-IBM platforms. IBM is committed to the open systems theme.

In the future, all of the products will continue to focus on improving availability and performance. In the DB2 arena, increased capacities and better price-performance will be provided in the future by coupling a collection of CMOS-based 390 microprocessors.

13. References

ACDL88 Anderson, M., Cole, R., Davidson, W., Lee, W., Passe, P., Ricard, G., Youngren, L. *Index Key Range Estimator*, **U.S. Patent 4,774,657**, IBM, 1988.

AnCo88 Anderson, M., Cole, R. *An Integrated Data Base*, In **IBM Application System/400 Technology**, Document Number SA21-9540, IBM, June 1988.

CEHH90 Crus, R., Engles, R., Haderle, D., Herron, H. *Method for Referential Constraint Enforcement in a Database Management System*, **U.S. Patent 4,947,320**, IBM, 1990.

ChGY81 Chamberlin, D., Gilbert, A., Yost, R. *A History of System R and SQL/Data System*, **Proc. 7th International Conference on Very Large Data Bases**, Cannes, September 1981.

CHHIM91 Cheng, J., Haderle, D., Hedges, R., Iyer, B., Messinger, T., Mohan, C., Wang, Y. *An Efficient Hybrid Join Algorithm: a DB2 Prototype*, **Proc. 7th International Conference on Data Engineering**, Kobe, April 1991.

ChMo93 Choy, D., Mohan, C. *An Efficient Indexing Method for Partitioned Data*, **IBM Research Report**, IBM Almaden Research Center, January 1993.

ChMy88 Chang, P.Y., Myre, W.W. *OS/2 EE Database Manager Overview and Technical Highlights*, **IBM Systems Journal**, Vol. 27, No. 2, 1988.

ClCo89 Clark, B.E., Corrigan, M.J. *Application System/400 Performance Characteristics*, **IBM Systems Journal**, Vol. 28, No. 3, 1989.

CLSW84 Cheng, J., Loosely, C., Shibamiya, A., Worthington, P. *IBM Database 2 Performance: Design, Implementation, and Tuning*, **IBM Systems Journal**, Vol. 23, No. 2, 1984.

CrHT90 Crus, R., Haderle, D., Teng, J. *Method for Minimizing Locking and Reading in a Segmented Storage Space*, **U.S. Patent 4,961,134**, IBM, October 1990.

Crus84 Crus, R. *Data Recovery in IBM Database 2*, **IBM Systems Journal**, Vol. 23, No. 2, 1984.

DHLPR89 DeLorme, D., Holm, M., Lee, W., Passe, P., Ricard, G., Timms, Jr., G., Youngren, L. *Database Index Journaling for Enhanced Recovery*, **U.S. Patent 4,819,156**, IBM, April 1989.

GMBLL81 Gray, J., McJones, P., Blasgen, M., Lindsay, B., Lorie, R., Price, T., Putzolu, F., Traiger, I. *The Recovery Manager of the System R Database Manager*, **ACM Computing Surveys**, Vol. 13, No. 2, June 1981.

IBM90 *Distributed Relational Database Architecture Reference*, **Document Number SC26-4651**, IBM, August 1990.

IBM92 *Systems Application Architecture Common Programming Interface Database Level 2 Reference*, **Document Number SC26-4798-01**, IBM, July 1992.

LeLi89 Lehman, T., Lindsay, B. *The Starburst Long Field Manager*, **Proc. 15th International Conference on Very Large Data Bases**, Amsterdam, August 1989.

LLPS91 Lohman, G., Lindsay, B., Pirahesh, H., Schiefer, B. *Extensions to Starburst: Objects, Types, Functions, and Rules*, **Communications of the ACM**, Vol. 34, Number 10, October 1991.

MBCS92 Mohan, C., Britton, K., Citron, A., Samaras, G. *Generalized Presumed Abort: Marrying Presumed Abort and SNA's LU 6.2 Commit Protocols*, **IBM Research Report RJ8684**, IBM Almaden Research Center, March 1992.

MHLPS92 Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., Schwarz, P. *ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging*, **ACM Transactions on Database Systems**, Vol. 17, No. 1, March 1992.

MHWC90 Mohan, C., Haderle, D., Wang, Y., Cheng, J. *Single Table Access Using Multiple Indexes: Optimization, Execution, and Concurrency Control Techniques*, **Proc. International Conference on Extending Data Base Technology**, Venice, March 1990.

Moha90a Mohan, C. *ARIES/KVL: A Key-Value Locking Method for Concurrency Control of Multiaction Transactions Operating on B-Tree Indexes*, **Proc. 16th International Conference on Very Large Data Bases**, Brisbane, August 1990.

Moha90b Mohan, C. *Commit_LSN: A Novel and Simple Method for Reducing Locking and Latching in Transaction Processing Systems*, **Proc. 16th International Conference on Very Large Data Bases**, Brisbane, August 1990.

Moha92 Mohan, C. *Interactions Between Query Optimization and Concurrency Control*, **Proc. 2nd International Workshop on Research Issues on Data Engineering: Transaction and Query Processing**, Tempe, February 1992. Also available as **IBM Research Report RJ8681**, IBM Almaden Research Center, March 1992.

MoLe92 Mohan, C., Levine, F. *ARIES/IM: An Efficient and High Concurrency Index Management Method Using Write-Ahead Logging*, **Proc. ACM SIGMOD International Conference on Management of Data**, San Diego, June 1992.

MoLO86 Mohan, C., Lindsay, B., Obermarck, R. *Transaction Management in the R* Distributed Data Base Management System*, **ACM Transactions on Database Systems**, Vol. 11, No. 4, December 1986.

MoNa91 Mohan, C., Narang, I. *Recovery and Coherency-Control Protocols for Fast Intersystem Page Transfer and Fine-Granularity Locking in a Shared Disks Transaction Environment*, **Proc. 17th International Conference on Very Large Data Bases**, Barcelona, September 1991.

MoNa93 Mohan, C., Narang, I. *An Efficient and Flexible Method for Archiving a Data Base*, **Proc. ACM SIGMOD International Conference on Management of Data**, Washington, D.C., May 1993.

TeGu84 Teng, J., Gumaer, R. *Managing IBM Database 2 Buffers to Maximize Performance*, **IBM Systems Journal**, Vol. 23, No. 2, 1984.