

DB2 UDB for OS/390: V6 Late Additions and V7 Enhancements

By Richard Yevich

The last year has seen many changes to DB2 for the OS/390, with enhancements to Version 5 that were retrofits from the Version 6 base. This year, there are many new enhancements for Version 6, some which are retrofits from Version 7. In addition, Version 7 has been announced and will probably be GA before the end of this year. To present exactly what will be discussed in this paper, the following lists are presented.

- DB2 UDB Server for OS/390 Version 6 Enhancements
 - Application enhancements
 - Support for defining identity columns
 - Savepoints
 - Ability to declare temporary tables
 - Update with subselect
 - Global transaction support
 - Columns in ORDER BY not in SELECT
 - Language support enhancements
 - SQL support for REXX
 - Stored procedures that are written entirely in SQL
 - Support for Java stored procedures (someday)
 - SQLJ, JDBC driver
 - Operational enhancements
 - Suspend update activity
 - Deferred definitions of data sets
 - DDF suspend and resume
 - Faster cancel thread
 - Data sharing improvements have been provided
 - New EDM pool parameter
 - New CHECKPAGE option during Image Copy
 - RUNSTATS enhancements
 - Performance enhancements
 - Star Join
 - Log I/O
 - Better optimization for complex queries
 - Volatile tables to use indexes are now available
 - Improved query parallelism has been implemented
 - Active log I/O performance improvement
 - Data Sharing improvements
 - Additional functional enhancements
 - Unicode client toleration support
 - IEEE float toleration
 - Controlling updates to partitioning key

- Toleration of separator differences
- New LANGUAGE bind options
- New operator for NOT
- DBPROTCL default change
- Instrumentation enhancements
- DB2 UDB for OS/390 Version 7
 - Application enhancements
 - Scrollable Cursors
 - Union in View
 - Row expressions
 - Limited Fetch
 - Unicode support
 - Commit & Rollback in stored procedures
 - Self Referencing Subselect for DELETES/UPDATES
 - Scalability enhancements
 - Unload utility
 - Parallel load
 - Improved optimization
 - More parallelism
 - Parallelism in Online REORG
 - Availability
 - Online ZPARM changes
 - More consistent restart
 - Online REORG changes
 - Online Load Resume
 - Checkpage option in copy
 - Management enhancements
 - Migration options
 - Utility lists
 - Utility parameters & Dynamic Allocation
 - Better constraint options
 - Statistics history
 - Precompiler services
 - Data sharing enhancements
 - Light restart
 - Control Center Enhancements
 - Index Advisor
 - DBADM create views for others
 - e-business enhancements
 - Improved DB2 Connect
 - Improved JDBC and ODBC
 - XML Support
 - Kerberos support

The list is extensive. Hidden within both the retrofits and the new features are a lot of application enhancements that will have a dramatic effect on not only how we design applications in the future, but on how we change existing applications when there is maintenance to be done. This is necessary and it is time that “yes, we fix it even if it is not broken” because there are many changes that will yield better performance. This has always been the case but most often was not done due to the fear that improving performance by using new features would cause additional problems. However, it is becoming much more important to realize that if an application can be changed, quickly and easily, to use new features to significantly reduce resource consumption, then the application is broken and needs to be “fixed”.

In the next two major sections, covering the primary details on the list of items, it will be strongly noted where major improvements in performance can be made. This will not only be in the sections called “performance enhancements”, but in the other sections also.

DB2 UDB for OS/390 Version 6 Enhancements

Application Enhancements

Support For Defining Identity Columns

There has been a need for an automatic sequence number column in DB2 for many years and it officially arrived as a new feature in Version 6. This column is called an IDENTITY column, a new phrase for declaring arithmetic columns (integer, fixed, etc.). This is a new column type for unique sequential number generation, with options for the underlying data type and the starting number as well. This will dramatically give better performance than the many and varied application controlled sequential number columns, with perhaps some exceptions of course (store-clock and its variants in certain situations). With a column being identified as an IDENTITY column, DB2 will generate a unique value starting with a set starting number and using an increment that are both user provided. For example:

```
CREATE TABLE ACCOUNT
(AACCOUNT INTEGER GENERATED ALWAYS
AS IDENTITY
(START WITH 100000
INCREMENT BY 100),
LAST_NAME CHAR(20),
...)
```

Another useful option, especially for performance and concurrency concerns where there are many concurrent inserters, is the ability to cache the next available sequences in memory. This is done with the CACHE (NO CACHE) Option. This tells DB2 whether or not to keep some pre-allocated values in memory, and can be used with the default value of 20, or set much higher up to the maximum integer value.

There are downsides to this feature are during a system failure all the cached values are lost, and the values will never be used since the high value has already been stored in the support catalog tables.

In a data sharing environment, each member will get its own range of values with caching. Therefore, if multiple members are requesting numbers from their own respective

caches, they will not be using consecutive numbers. This will be a problem if an absolute clustering sequence is a high priority requirement.

It would be wise to use the NO CACHE option if you truly need to guarantee clustering order with ascending sequence numbers.

Using sequence numbers always causes a high probability of 50/50 index page splits, especially if CACHE is greater than 1. This is more of a maintenance problem, but successive inserts will more than likely not be in order.

If the IDENTITY columns are defined with the GENERATED ALWAYS option, the columns are not updateable, and this will have implications when loading data.

In addition, there is only one identity column allowed per table and this feature does not allow nulls. Other restrictions are that the table cannot have either a fieldproc or an editproc defined on it. The editproc is the more significant problem, as there are still tables being compressed with non-DB2 compression, which generally is implemented with an editproc. More restrictions are the use of WITH DEFAULT is not allowed and Global Temporary Tables cannot have identity columns (but Declared Global Temporary Tables can). If a table is created using CREATE LIKE from a table with an identity column, then the attributes are not inherited unless you use the INCLUDING IDENTITY COLUMN ATTRIBUTES clause.

If the application is using multiple insert transactions on the same table, then it is impossible to guarantee the sequence of the inserts. If the sequence is important, then it might become critical to use exclusive locks. If a transaction is rolled back, then the allocated numbers are not used. If recovery is to a prior point-in-time, any new numbers used will start from highest allocated. There are also some restrictions when using views.

There are special catalog tables to control this feature, and there are changes to others. They are as follows:

- SYSIBM.SYSSEQUENCES
 - One row each identity column
 - Definition fields
 - Last Assigned Value
- SYSIBM.SYSSEQ2
 - One row for each identity column
 - Release dependency and dependent table indicators
- New Column values for DEFAULT in SYSCOLUMNS
 - I = GENERATED ALWAYS
 - J = GENERATED BY DEFAULT

Quite often, there is a need to know the number that was used on the last insert, especially if it is going to be used in other related tables. If you need to know the last column number generated, there the new scalar function needs to be used as follows:

```
SET :HV = IDENTITY_VAL_LOCAL()
Result is DECIMAL(31,0)
```

If a commit or rollback occurs after the insert, a null will be returned from this function, and it also only works for single inserts. It is recommended to store the value after an insert and check the return code carefully.

This is one area where we will have additional information posted in the “Hints and Tips” sections, because there is much analysis going on at this time, at several locations. The reason is that there are a lot of little drawbacks to a normal use of these identity columns. As we get all the information, and have a list of workarounds to the problems, it will be posted on the site.

Savepoints

Savepoints have been around for years in other database management packages. They allow for bookmarks (milestones) to be made between the start of a unit-of-recovery and the end of a unit-of-recovery (some call these units-of-work but it truly an incorrect definition). An application can set a savepoint within a transaction. Once a savepoint has been set, application logic can undo the data and schema changes made back to the savepoint. The application sets the savepoint without affecting the overall outcome of the transaction.

Using savepoints makes coding applications more efficient. You no longer need to include so much contingency logic in your applications. Multiple savepoints can be set, and this would be a big benefit to a lot of back-end transactions in support of e-business. A transaction could be underway, in a dynamic web environment. Often a user wants to backtrack some of the way, but not all. Using savepoints can help greatly, and allows for several bookmarks to be placed during e-commerce transactions.

So basically this new function supports external savepoints, which enables milestones within a unit-of-recovery, representing the state of the data and schema at a particular point in time. Once these have been set, an application can use ROLLBACK to restore to a savepoint. This means that the application/transaction has reached a point during a UNIT-OF-WORK and there is a need to back out to some point, without performing a ROLLBACK over the entire UNIT-OF-WORK.

Since there can be multiple SAVEPOINTS, part of the enhancement allows for naming the individual SAVEPOINTS. Then a ROLLBACK can be used to back out to any point that is required based on the dynamics of the transaction. It is even allows to leap over individual savepoints. For example:

```
Code...
SAVEPOINT AAA
SELECT ...
Code...
SAVEPOINT BBB
Code...
ROLLBACK TO SAVEPOINT AA
```

The syntax required to rollback to a savepoint is the ROLLBACK TO SAVEPOINT xxxx. This would back out all data and schema changes made after the savepoint. However, there are actions that will not be rolled back:

- Updates outside local DBMS (remote DB2s, VSAM, CICS, IMS)
- Changes made to global temporary tables, however declared global temporary tables changes will be backed out.
- Opening/closing cursors
 - Can optionally use ON ROLLBACK RETAIN CURSORS

- Cursor positioning
 - Can optionally use ON ROLLBACK RETAIN CURSORS
- Locks acquired/released
 - Can optionally use ON ROLLBACK RETAIN LOCKS
- Caching of rollback statements

There are other restrictions, as savepoints cannot be used with the following:

- Global transactions
- Triggers
- User-defined functions
- Stored Procedures
- UDFS
- Triggers nested within UDFs

Declared Global Temporary Tables

This is the ability to declare a temporary table in a program or a transaction, as required and not have it go away at a commit. Declared temporary tables (DTT) complement the existing global temporary tables made available in Version 5 of DB2 for OS/390. But there are major differences. Declared temporary tables do not have descriptions in the catalog tables. These new tables also support indexes, UPDATE statements, and positioned DELETE statements. It is also possible to implicitly define the columns and use the result table from a SELECT. They can be used as a way to temporarily hold or sort data within a program, especially in support of ROLAP and MOLAP queries for warehouses or warehouse tools. They can be used as a staging area for IMS or VSAM data so it is SQL and ODBC accessible.

The tables will be qualified by the name SESSION and this can be named explicitly in the table name, which is a good idea (always hate to leave the defaults up to DB2). The name SESSION can be in the QUALIFIER BIND option on the plan/package.

Declared Temporary Tables may be defined just like any other table, with column definitions, but in application code, there are a couple of easier ways to do this.

```

DECLARE GLOBAL TEMPORARY TABLE SESSION.ACCOUNT
    LIKE CLAIMS.ACCOUNT
INSERT INTO SESSION.ACCOUNT
    SELECT * FROM CLAIMS.ACCOUNT
    NOT LOGGED

DECLARE GLOBAL TEMPORARY TABLE SESSION.ACCOUNT
    AS
    (SELECT * FROM CLAIMS.ACCOUNT)
DEFINITION ONLY
NOT LOGGED

```

There is some minor logging overhead but only Undo records are logged, since declared temp tables participate in rollbacks to defined savepoints within a unit-of-recovery. The

application can perform the full range of DML (INSERT, UPDATE, SELECT, DELETE) on the data in the table, and it will be supported by rollback to savepoint or to the last commit point. The table does exist until thread termination, or if using thread reuse, it will exist until it is implicitly dropped.

There are no locks taken (PAGE, ROW, TABLE) on the declared temporary table, as it is only known by the thread and it is not externalized. However, there are locks taken on tablespace and DBD but only share mode. Since these tables persist, there use does not require a declared cursor with hold, to hold rows across commits.

These declared temporary tables are not materialized in DSNDB07, which is a good thing. There are materialized in a segmented TEMP tablespace, which requires some additional planning as there can be one, or several, and they need to be sized correctly.

There are however, performance implications when using these tables, even with them not being materialized in DSNDB07. Static SQL referencing a DTT will be incrementally bound at run time and the costs are equivalent to using dynamic SQL. Therefore high transaction applications need careful evaluation. Also, the table will need to be explicitly dropped when no longer used. But when it is dropped, it will not invalidate plans/packages, but will help thread reuse and DBD management. Dynamic SQL statements referencing a DTT will not use the cache.

Any indexes created on these declared temporary tables must be qualified with SESSION also. Some basic statistics are kept and maintained by DB2 for these indexes. These statistics are used for dynamic optimization, but they are not stored in the catalog.

Even those these are not real tables according to the normal definition; their pages can still go to the LPL. To remove these pages, it will be necessary to use the START DATABASE command.

It will be necessary to create the database AS TEMP. This database is not sharable across data sharing members; therefore use member must have its own created. It would be better to create several segmented tablespace and let DB2 decide where the tables go. The same as with DSNDB07, it is recommended to make each the same size and then spread them across volumes. They need to be sized accordingly to accommodate growth and concurrent transactions, for ALL DTTs at a point-in-time.

There are some more tablespace considerations for these TEMP tablespaces. Tables cannot span tablespaces so it may be necessary to use small tablespaces to control the size of declared tables. And there may be a need to have different tablespaces for 4K, 8K, 16K and 32K page sizes. DB2 will choose appropriate size and if not there the declare will fail. The PUBLIC will have the authority to create DTTs in these tablespaces. If this impact is unknown, it may cause EDM pool storage increases to be necessary, since the size of this DBD is limited to 25% of the pool. The only utility allowed against DTTs is the REPAIR DBD.

There are other restrictions on the use of declared temp tables:

- No LOBS or ROWIDs
- No RI
- Cannot be used in a CREATE TABLE LIKE
- No Sysplex Query Parallelism
- No Dynamic SQL Caching
- No support for ODBC/JDBC functions that rely on the catalog definitions
- Thread reuse not supported for DDF pool threads

- Cannot be used within triggers

Update With Subselect

Now finally, it is possible to use a subselect to determine the value that is to be used in the SET clause of a normal UPDATE statement, but not in the SET clause of transition data in a trigger. An update can be performed based on criteria from another table. This feature can be used for both searched and positioned updates. But as with many enhancements, there are usage restrictions:

- The number of columns selected will have to equal the number of columns updates
 - Data types must also be compatible
- The subselect must return a single row
- The subselect must reference a table, view, synonym, alias or join
- GROUP BY or HAVING cannot be used

If no rows returned from subselect, a null will be assigned to the column value for the update (-407 if nulls not allowed).

This feature is ideal for updating a column based on result of column functions but it cannot be self-referencing (until Version 7)

```
UPDATE EMP
SET SALARY = (SELECT AVG(SALARY)
              FROM DEPT
              WHERE SALGDE = 'MGT100')
WHERE EMPCLASS = 'PEON100';
```

Global Transactions

Global transaction support has been added allowing applications to take advantage of global transaction features. DB2 units of recovery can share locks and access the same data when they are part of the same global unit of work. A sync point manager, such as Component Broker or IMS, must coordinate commit operations using a two-phase commit protocol.

Global transactions allow for several units-of-work to share locks if they are all participants in a unit of work. The large unit of work is referred to as a global transaction. This provides a way to avoid deadlocks between transactions. Many installations may want to combine transactions into a large transaction (single unit of work) for several reasons:

- Example: adding OO interfaces on existing transactions

Global transactions are simply an extension of the distributed unit of work, adding the ability to share locks. These also support IMS, RRS and DDF transaction by using a new token, the global transaction ID, or XID.

All URs in a global transaction must run on same subsystem. Each part can see uncommitted updates of other parts and the sharing of locks is limited to normal transaction locks. The LOCK TABLE statement or use of a partition key update could cause deadlocks and

timeouts. These global transactions require many re-design issues to be considered along with the risk and exposure issues.

Columns In ORDER BY Not In SELECT

The title says it all, and this is a BIG performance impact in many applications. Quite often the columns required to do the ordering are not required to be in the result set. Prior to this enhancement, they had to be. This increased the row length being sorted, and increased the amount of data being returned to the programs. This is one of those features that needs to be communicated to all application developers so that offending SQL can be corrected during routine maintenance.

Language Support Enhancements

SQL Support For REXX

SQL statements can now be issued from REXX programs, and can be anywhere a REXX command can be. It is also possible to write stored procedures using REXX. Almost all SQL statements that DB2 for OS/390 supports are allowed in REXX programs.

Stored Procedures That Are Written Entirely In SQL

One of the most important enhancements is the support for a new stored procedure programming language, called SQL procedures. These are SQL-only stored procedures based on the ISO/ANSI standard SQL/PSM. It is possible to port the same stored procedure to any member of the DB2 family and simplifies the process of migrating stored procedures between other DBMSs and the DB2 family.

The implementation of SQL stored procedures is based on the SQL standard, and supports constructs that are common to most programming languages. This is part of what has been known as SQL 3 -- now called SQL 99. DB2 has adopted this SQL Procedures language from the SQL standard. It is a standard compliant stored procedure language rather than proprietary, as is the case with Oracle, Sybase, and SQL Server.

It allows for stored procedures consisting entirely of SQL statements and is supported across all members of the DB2 family. SQL Procedures functionality provides the benefit of writing stored procedures in a standard portable language.

A SQL stored procedure consists of a CREATE PROCEDURE statement to define the procedure and single or compound SQL statements. A compound SQL statement can include declarations of variables, conditions, cursors, handlers, flow control statements, assignment statements and traditional SQL for defining and manipulating data.

The following stored database language capabilities are supported:

- Statements to direct flow control:
 - CASE statement
 - GOTO statement
 - IF statement
 - LEAVE statement
 - LOOP statement

- REPEAT statement
- WHILE statement
- Compound statement:
- Assignment statement
 - SET
 - GET DIAGNOSTICS
- Specification of condition handlers:
 - Condition handlers tell the SQL stored procedure what to do when an SQL error or an SQL warning occurs, or when no more rows are returned from a query.
- Specification of statements to signal and resignal conditions:
 - SIGNAL
 - RESIGNAL (on Workstation only)
- Declaration of SQL local variables
 - DECLARE.....DEFAULT.....
- SQL statements

The following chart, from IBM, shows both the similarities and differences in the SQL Procedures Language features that have been delivered on each of the IBM platforms.

Functional Item	DB2 for UNIX, Windows, OS/2	DB2 for OS/390
Savepoint Support	Supported	Supported
SIGNAL	Supported	Supported
Statement Size Limit	64K	32K
Dynamic SQL Authorizations	Default = executor's authority	Default = executor's authority
Dynamicrules BIND (chng above static model)	Supported	Supported
Nested NOT ATOMIC Compound Statement	Supported	Not Supported
Nested ATOMIC Compound Statement	Not Supported	Not Supported
RESIGNAL statement	Supported	Not Supported
GRANT/REVOKE statement on SQL Procedure	Not Supported	Supported
Nested Stored Procedure Calls	Supported	Supported
Dynamic calls	Supported	Supported
Multi-rowed result sets	Supported	Supported
COMMIT/ROLLBACK statements	Supported	Supported
CONNECT Statement	Supported	Supported
Standalone SQLCODE/SQLSTATE	Supported	Supported

CREATE PROCEDURE statement	Not Supported	Supported
Static DDL	Supported (via VALIDATE RUN)	Supported
Allow colon in front of host variables	Not Supported	Not Supported
Single Statement Procedure	Supported	Supported
ITERATE Statement	Supported	Not Supported
GOTO (non standard)	Supported	Supported
C Comment	Supported	Not Supported; PSM comment supported only
Overriding PREP and Compile Options	Supported	Supported
Length of Parameter names	128 Bytes	128 Bytes
Maximum Length of Character Variables	255 for VARCHAR; 32K for LONG VARCHAR	255 Bytes
Date Arithmetic	Supported	Supported
SET Assignment statement	Supported	Supported
SELECT statement on right side of SET	Supported	Supported
RETURN	Supported	Supported

You can use the Stored Procedure Builder to build SQL stored procedures. The Stored Procedure Builder is part of the DB2 Management Tools Package. It is a graphical windows application that runs on Windows, AIX, and Solaris supporting development of stored procedures written in Java or the SQL Procedure Language. It can work as a Microsoft Visual Studio plug-in, or with Microsoft Visual Basic, IBM VisualAge for Java, or stand-alone.

Operational Enhancements

Suspend Update Activity

This adds the ability to suspend DB2 logging through the use of two new options added to the -SET LOG command: SUSPEND and RESUME. These are used to temporarily freeze logging so a consistent instantaneous copy of the data can be made, and are intended for use with RVA Snapshot or ESS Flash Copy. It is essential to issue this command before taking these types of copies for offsite recovery.

A checkpoint is taken during a suspend and log buffers are externalized. The BSDS get updates, a latch is taken on log to prevent any new updates, and pending writes are not externalized from the buffer pools. When logging is resumed, the latch is released and logging of updates is allowed.

Deferred Definitions Of Data Sets

This adds the ability to defer the definition of datasets. SAP for example, will not have to create all those unused objects, which will allow for a faster install. The normal time would be .5-.6 seconds to perform define of a dataset. To allow this, there is a new option on Tablespace/Indexspace, which is DEFINE NO. With this, no ICF entry made and no datasets are created in the VVDS or the VTOC. This yields better space management and provides for DASD savings. The objects are recorded in the DB2 catalog, but have a -1 recorded in the SPACE column.

When a deferred define object is accessed, the application will get +100. When a write occurs (LOAD or Insert), DB2 resets the status and creates the dataset.

Be aware, and make DASD folks aware, that these datasets may one day be created, and therefore the need exists to be sure sufficient space is available. Placement of the datasets may be a performance and maintenance issue for delayed defines.

DDF Suspend And Resume

DDF Suspend and Resume support has been added for DDL. This adds new function to DDF STOP command to allow for a customer to quiesce DDF processing activity and release all locks held by DDF server threads. There is also a new option to suspend DDF server processing without termination.

Prior to V6, DDF DRDA server threads may hold locks that prevent database maintenance, especially with running the new DDF pool threading.

Faster Cancel Thread

Prior to this enhancement, thread cancellation was dependent on thread status. The thread had to be active. If inactive, then resources were held and this could cause problems. STOP DB2 commands also waited.

Now, even if the thread is inactive it can be cancelled. During the next request, the application will discover the cancellation. This is only applied to local threads and will not have effect on must complete functions of Commit and Rollback.

Data Sharing Improvements Have Been Provided

Enhancements have been made so that a member can be stopped without doing CASTOUT operations. The P-locks will still be retained, with IX mode for objects, in which the member was last updater. This will prevent utilities, but allow other members to update.

The GBP will be in a 'failed persistent' state for member shut down. This improves shutdown performance, about 40% in a 2-way environment (range 15-80%). There is no observed increase for restart time.

This is a feature that you do not want to use for all members, as data may be inconsistent because the latest version may be in GBP. This would require a group restart.

New EDM Pool Parameter

A new DSNZPARM for the EDM Pool, called EDMBFIT, has been added to help with virtual storage constraints. This is for EDM pools larger than 40 MB to provide relief when

loading large objects. The algorithm for searching free chain is changed. There are two values for the ZPARM, YES and NO.

YES is for the better-fit algorithm. This will search the free chain for best place for objects. It does require more latches and this may increase latch suspension times. However, it does provide better management of storage.

No is for the first fit algorithm. This places objects in the first available free space with less latching. This should be used unless you are constrained in DBM1.

New CHECKPAGE Option During Image Copy

The CHECKPAGE option has been added to Image Copy for additional validity checking during image copies. This is for tablespaces or indexspaces and applies to all objects in the list. Any errors are reported immediately. There is about a 5% overhead for an index, 15% for a tablespace.

RUNSTATS Enhancements

There are now more statistics for the optimizer. Now RUNSTATS collects statistics for SYSCOLDIST, the non-uniform statistics. This will provide for better access path selection but will potentially increase the catalog size.

Also, RUNSTATS will collect additional space statistics. These statistics collected are to help estimate the number of extents.

Performance Enhancements

Star Join

This is an enhancement of a technique that has been in DB2 for many many years, only it was not called a Star Join. But it was always behind the scenes. Now it is not only brought out in front, but is renamed. It is a join method that is in support of a Star Schema and its variations.

A Star Schema is generally described as a central fact table surrounded by several dimension tables. This is a particular type of design called a multi-dimensional model. The fact table keys hold the keys of each dimensional table. This kind of processing normally requires many joins. DB2 uses the “early cartesian product” technique. DB2 first joins the dimension tables, which generally have no join predicates, as they are joined to the Fact table. Through V5, DB2 was limited to 5 tables and V6 increases the limit to 6 tables. This is different from the 225-table join limit for V6 as this is only for the Cartesian join method.

This join can be seen in more places than just a Star Schema. It appears anywhere there are two or more small tables being joined to a very large table. It can also be forced by a series of full outer joins over the small tables or dimension tables, and then a right outer join to the fact or large table.

Basically, the “Preferred Cartesian Join” becomes the Star Join and becomes identified in PLAN_TABLE. This is a nice change since the Early Cartesian Join was never identified and was only inferred by joining small tables where there were no join columns in the SQL.

Log I/O

Reading of active log was not originally designed for diverse and frequent use. But due to new requirements (data sharing, online REORG, ISV products, DROPR etc), this has changed. Now direct reads can go against the secondary logs (when > 3 concurrent). Certainly this requires dual logging. There is improved I/O response time for reads/writes.

Volatile Tables To Use Indexes Are Now Available

This enhancement is helpful for tables that start small when RUNSTATS is executed, but grow during execution. The problem was that the optimizer normally would choose a tablespace access path. Now there is a new DSNZPARM called NPGTHRS to help in this situation. It has 3 possibilities for setting:

- 0 = (default) standard cost optimization
- n = NPAGES < n use preferred index
- -1 = (not recommended) always use preferred index

If there are no statistics at all (-1), the NPAGES is seen as 501 by the optimizer. In this case therefore, it is not recommended that you use anything less than 502. You should be extremely careful with this parameter since it could easily degrade performance if set wrong by forcing index access where scans would be more efficient. The problem as I see it, is this is a SYSTEM WIDE parameter, and not an application or table parameter.

Improved Query Parallelism

It was always possible for parallelism to select a number of degrees that could not be satisfied based on the resources available. Now, the DSNZPARM PARAMDEG can control the maximum degree. It was there before, but has now been externalized on install panels.

Data Sharing Improvements

There have been insert performance improvements for data sharing by the reduction in index P-Locks. Now the system waits for possible re-acquisition of the lock before releasing the lock. For MEMBER CLUSTER tablespaces, the P-Locks are held past commits assuming the same member may update again. There is a lessening of write I/Os for frequently modified pages (better queue maintenance).

CLOSE YES is no longer required to close non-GBP dependent datasets. This could impact the DSMAX setting. Now, CLOSE NO objects are done in LRU sequence. Name Class Queue Support has been added but requires CF Level 7. Class queues will be organized by DBID, PSID and partition number (IFCID 263 enhanced to support).

There is now better tracing for asynchronous requests as IFCID 329 tracks wait time for GBP requests, converted from sync to async.

Additional Functional Enhancements

Unicode Client Toleration Support

This feature requires LE for OS/390 V2 R9 in order to be operative. Now, for a character conversion, rather than just search SYSSTRINGS for a translation table, DB2 will first search the cache, then SYSSTRINGS, then LE. This is a requirement for client Unicode support. The rest of the Unicode world will appear in V7.

IEEE Float Toleration

Db2 now tolerates the IEEE Binary Floating Point host variables as well as the normal S/390 HFP format. This is only for ASM, C, and C++ languages when they specify the FLOAT (IEEE) option for precompiles and compiles. DB2 will convert the IEEE format to the HFP format.

Controlling Updates To Partitioning Key

It all began in V5, when updating of partitioning keys was added. This is a critical requirement, but not really for DB2 OS/390 written application. Rather it is for applications that never knew the data was going to be partitioned. This would include applications that migrated to DB2 from somewhere else, and applications whose tables have reached some threshold that they now want to partition them without changing all the underlying programs.

This enhancement adds a new DSNPARM called PARTKEYU, which has three values. “YES” means to do what it was already doing – allow the updates, but drain the from:to partitions, draining the partitioning indexes from:to, and draining all the non-partitioned indexes. “SAME” would allow updates as long as they did not cross partition boundaries. This would generally be where the higher level columns/data of the partitioning key, did not change. If an update did cross partitions, then a –904 would occur. “NO” simply says to return a –151 and prevent all partitioning key updates.

Toleration Of Separator Differences

This just allows dynamic SQL to use a decimal point separator, instead of the comma.

New LANGUAGE Bind Options

Private protocol is going away and the movement to DRDA as a replacement has begun. However, there are some COBOL programs that use hyphens in cursor names. A new bind option, called LANGUAGE will allow this.

New Operator For NOT

Other databases have used different operators for NOT from what was enabled on the OS/390. While <> is generally the accepted one (as well as NOT), !=, !> and !< were already in use, and many of these systems are migrating to OS/390, it now allows them.

DBPROTCL Default Change

DBPROTCL is a new DSNZPARM in V6, to assist in the movement to DRDA from private protocol. However, its default was set to DRDA and this did cause some problems. This enhancement changes the default to PRIVATE, and requires setting it to DRDA when required.

Instrumentation Enhancements

Dynamic statement cache statistics are now collected for every statement in the cache, reporting on accumulated CPU and wait time, RID list failure, and when the statistics interval began.

IFCID 23, 24, and 25 have been changed to collect utility sub task counts.

DB2 UDB for OS/390 Version 7 Enhancements

Application Enhancements

Scrollable Cursors

Scrolling has always presented problems both for the application programmer and for proper performance, since there was no real perfect solution to every requirement. One could scroll forward by walking an index or walking through a result set. This really was not too hard, just cumbersome. However, scrolling backwards is much more difficult! When scrolling forward, the only effective way from a performance perspective was to use an ascending index. If it was required to scroll in reverse, it either required some sophisticated programming or another index, this time a descending one.

Now we will have the use of scrollable cursors and can move more of the work to perform this operation into SQL. Some of the features that can be used with this technique are:

- Fetch the next row
- Fetch the previous (prior) row
- Fetch the first row
- Fetch the last row
- Fetch the current row
- Move the cursor forward or backward a given number of rows
- Use either relative or absolute positioning in the result set
- Have the data refreshed or not

Scrollable cursors are defined using parameters on the DECLARE CURSOR statement. The following statements show the new cursor types in V7:

```
DECLARE cur1 INSENSITIVE SCROLL CURSOR FOR ...  
DECLARE cur2 SENSITIVE STATIC SCROLL CURSOR FOR ...
```

Cursors can be defined as sensitive to changes in the data, or insensitive. Sensitive static cursors will show changes made by the user and other users, except inserts, while insensitive scroll cursors are not updateable. Regardless, for the result set both types use a fixed declared

temporary table created automatically by DB2. Static scroll cursors require a declared temporary database (TEMP) and predefined segmented table spaces.

While sensitive STATIC cursors use a declared temporary table, sensitive DYNAMIC cursors do not and directly access the base table. DB2 V7 on OS/390 only supports sensitive STATIC cursors. Static scroll cursors will have their result table kept in sync with the base table as each row is fetched. There will be a fixed number of rows in the result table, and it will be dropped when the cursor is closed.

The following example shows an inquiry to a frequent flyer database from a customer checking their last 5 air transactions.

```
DECLARE MILES STATIC SENSITIVE SCROLL CURSOR FOR
  SELECT TRIPDATE, MILES, DESTINATION
  FROM PLUS
  ORDER BY FLYDATE ;
OPEN MILECURS;
FETCH LAST MILECURS;
DO I= 1 TO 5;
FETCH PRIOR MILECURS INTO :FLYDATE, :MILES, :DESCRIPTION;
  Display logic..
END;
```

Union Everywhere

UNION and UNION ALL will now be supported everywhere. UNION in views and table expressions are two of the most desired features, but it is UNION is predicates, inserts, and updates that offer the most power. Another one of the most requested features is to allow UNIONS in subqueries, especially in support of some of the adhoc report writing tools. UNION everywhere means that now UNIONS can be included in subqueries, and all other types of predicates also. For example:

```
SELECT Z.A, Z.B, Z.C
FROM (SELECT X, Y, Z
      FROM XX
      UNION
      SELECT D, E, F
      FROM YY) AS Z(A, B, C)
WHERE Z.A = :hv1
AND Z.B IN (SELECT QW
           FROM ASDF
           UNION
           SELECT ER
           FROM LKKH)
AND (Z.A, Z.C) IN (SELECT ASDF, QWER
                  FROM TUIO
                  UNION
                  SELECT TRET, WERE
```

```
FROM PIOUS)
AND (Z.A, Z.C) <> (:hv_from, :hv_true)
```

This feature has been allowed in DB2 on the other platforms for several versions, and not finally it is available on the OS/390 platform. Over the last several years, the need for UNION statements has diminished somewhat with the additions of outer joins and the CASE expression. But some other features that will be added in the future will require not only UNION statements, but also UNION statements defined in views. The whole concept of using common table expressions for recursive SQL will require this in order to be implemented.

Row Expressions

Row expressions are an extension to predicates. They allow more than one set of comparisons in a single predicate using a set of values is equal or not equal to a set of values, or a set of values in a set of columns. A subquery that returns more than one column and even more than one row could be used to represent a set of values for the comparison. This predicate in the example below will be true when all three columns on the left equal any three values in any single row returned in the result set from the subquery. This will also allow the use of quantified predicates with row expressions. In the previous example for UNION everywhere, there were other examples of using set comparisons.

```
SELECT *
FROM TABLE
WHERE (col1, col2, col3) IN (SELECT cola, colb, colc
                             from TABLE)
```

Limited Fetch or FETCH FIRST

Limited fetch gives us the ability to tell DB2 to only fetch the first or top 'n' rows from a result set. There are ways in SQL and the application program to do this today, but they are very inefficient generally. Limited fetch adds a new clause to provide this functionality with improved performance. The clause sets a maximum number of rows that can be retrieved through a cursor, which could be limited to even just 1 row. The application now has a way to tell DB2 that it does not want to retrieve more than some fixed number of rows regardless of how many rows qualified for the result set.

```
FETCH FIRST n ROWS ONLY
```

The above stops processing after 'n' rows. This provides direct control to the application programmer on the size of the result table. If an application tries to fetch more than the 'n' rows, it will receive the SQLCODE of +100, or end of set. The use of this also defaults to OPTIMIZE FOR 'n' ROWS if the OPTIMIZE FOR clause is not specified. If the OPTIMIZE FOR clause is specified, the small number of rows specified between the 2 clauses is used for both optimization of the query and for the size of the communication buffer.

Unicode Support

Unicode is an encoding scheme that is able to represent codepoints or characters of many different languages and geographies. The IBM implementation is an implementation of the ISO-10646 standard. It supports both single byte and double byte Unicode and will help support data across multinational boundaries. It will allow for the representation of codepoints and characters of virtually all languages.

The Unicode character-encoding standard is a variable length character-encoding scheme supporting multiple encoding schemes includes characters from most of the world's languages. There are new options on CREATE and ALTER table to support this addition to the CCSID selections.

Commit & Rollback In Stored Procedures

If I could venture one word on this feature, it would be "Ouch!" In the first instantiation of stored procedures there was no COMMIT at all. Then came the ability to COMMIT ON RETURN to reduce the network traffic. Now V7 gives us the ability to commit or rollback in stored procedures, as long as they are not nested where this would be ignored. Using either of these will commit/rollback entire unit of work including changes made from the client before the stored procedure was called. This will have implications only on new stored procedures or on careful enhancements to existing stored procedures. COMMIT and ROLLBACK cannot be used:

- For User Defined Functions,
- If client is using 2-phase commit
- Nested Stored Procedures

Self Referencing Subselect for DELETES/UPDATES

In prior releases, a searched update or delete could not contain a where clause referring to the same table. If it was tired, then there would be a SQLCODE of -118 returned. Now in Version 7 *searched* updates and deletes can now self-reference. The subquery will be completely evaluated before an UPDATE or a DELETE occurs. A non-correlated subquery is executed only once before an update or delete. However, a correlated subquery may require two steps. First it would create a workfile with new values or RID to delete, and second read records from work files and perform update or delete.

```
UPDATE EMPLOYEE A SET SALARY = SALARY + 10000
WHERE SALARY < (SELECT AVG(SALARY) FROM EMPLOYEE B)
WHERE A.DEPT = B.DEPT
```

Scalability Improvements

Unload Utility

One of the new utilities delivered is the new unload utility. This will deliver better performance and more flexibility than DSNTIAUL and will support the following:

- Both Tablespace and Image Copy as the Unload source
- UNLOAD-ing of multiple partitions in parallel
- Field selection
- Ordering and formatting options via an SQL-like syntax

Parallel Load

There is now support for parallelism during the LOAD utility. This is a nice improvement in terms of dealing with short windows in which to load a lot of data. This has especially been a problem where NPIs were involved, causing us to have to drop and recreate the NPIs in order to get the loads done. Now it is possible to submit a single job with several input files to be loaded in parallel. The performance is much faster and the contention on the NPI is eliminated. The number of parallel load tasks will be determined by the number of CPUs and virtual storage available, and by the number of threads available.

```
LOAD INTO TABLE tab1 PART 1 INDDN infile1
      INTO TABLE tab1 PART 2 INDDN infile2
...
```

Improved Optimization

As with every new release, there are improvements to the way SQL statements are optimized and/or rewritten. V7 will provide for indexable correlated subqueries (this is VERY significant for performance).

There is also improved sort avoidance with ORDER BYs where columns for the ordering do not have to be selected.

And another major impact enhancement is the support for backwards scanning of indexes, in particular when using the MAX or MIN functions. Prior to this improvement you would possibly need two indexes, an ASCending index and a DESCending index for best performance for these functions. With this enhancement, either an ascending or descending index can be used for either MIN or MAX functions. This is not due to the bi-directional index that we have been hearing about as a possible enhancement, but is in reality a backwards-scrolling technique supposedly. A little guess work here.

More Parallelism

Parallel processing is now supported for IN-list index access. Prior to Version 7, DB2 could only use parallelism when the in list access was for the inner table of a parallel group.

Now there is full exploitation of parallelism for In-lists. This is a big improvement over sequential access and will be reflected in the PLAN_TABLE as well. The PARALLELISM_MODE will be set and the ACCESS_DEGREE will be 'N'.

```
SELECT COLA, COLB
FROM EXAMPLE_TABLE
WHERE COLA IN (10,20,30,40)
If there is an index on COLA, parallelism
will be used for access
```

Parallelism in Online REORG

There is an improvement in the elapsed time in the BUILD2 phase of REORG, for both SHRLEVEL CHANGE and REFERENCE. This will be seen when reorging one partition or a range of partitions. The logical partitions of the NPI will be updated in parallel. This makes on-line reorg for partitions less disruptive and is less of an outage. It will dispatch multiple subtasks to process the logical partitions asynchronously where before the BUILD2 phase process was synchronous. With this feature, each subtask will be assigned to perform updates to a logical partition. The number of subtasks can be constrained by a lack of virtual memory, insufficient number of available threads, or insufficient processors.

The -DISPLAY UTIL command will show number of records updated for all logical parts. There is a new DSNU message issued during the BUILD2 phase of the REORG utility that indicates the number of logical partitions updated in parallel.

```
DSNU1114I      LOGICAL PARTITIONS WILL BE LOADED IN PARALLEL,
                NUMBER OF TASKS - nnnn
```

Availability Improvements

Online ZPARM Changes

Version 7 will soon allow for up to approximately 60 of the most popular DSNZPARMS to be dynamically changed. This is a subset of the DSNZPARMS. The advantages are that we can set values and tailor parameters to the current workload with any outage. The ZPARM member is changed dynamically in its entirety by activating a different ZPARM member, through the use of a new command for changing parameter module.

```
-SET SYSPARM LOAD (modname)
    Loads the named parameter module-default
    (if not specified)=DSNZPARM

-SET SYSPARM RELOAD
    Last named subsystem parameter module is loaded into storage

-SET SYSPARM STARTUP
    Loads the initial parameters from DB2 startup
```

RLFERRD	TSTAMP	MAXDBAT	URCHKTH	IRLMSWT
BLKSIZE	UNIT2	DSSTIME	TBSBPOOL	ABIND
PRIQTY	ARCRETN	STATIME	IDXBPOOL	ABEXP
SECQTY	QUIESCE	DBPROTCL	WLMENV	BINDNV
UNIT	ARCWRTC	PTASKROL	PTASKROL	DSMAX
ARCPFX1	ARC2FRST	RLFAUTH	CONDBAT	NUMLKTS
ARCPFX2	MAXRTU	RLFTBL	AUTHCACH	CDSSRDEF
CATALOG	DEALLCT	RLFERR	EDMPOOL	NUMLKUS
ALCUNIT	LOGLOAD	PCLOSEN	EDMSPAC	RECALLD
PROTECT	IDBACK	DLDFREQ	EDMBFIT	UTIMOUT
ARCWTOR	IDFORE	STORMXAB	MAXRBLK	BMPTOUT
COMPACT	CTHREAD	STORTIME	MINRBLK	DLITOUT
RRULOCK	SEQCACH	SEQPRES	DESCSTAT	RETLWAIT
RELCURHL	OPTHINTS	RETVLCKF	CONTSTOR	DBSCRWV

More Consistent Restart

More consistent restart allows for the ability to cancel the thread and recover without having to wait for rollbacks of long running jobs to complete. A new command will allow long-running jobs or statements that have not issued commits to be removed quickly from the system.

```
-CANCEL THREAD ... (NOBACKOUT)
```

Objects will be inconsistent after the cancellation and will be placed in a recover pending state. It will be necessary to recover the table space and indexes.

Online REORG Changes

On-line REORG now has a FASTSWITCH keyword. This replaces the approximately 3-second outage associated with the renaming of original and shadow dataset copies. It is a memory-speed switch of the MVS catalog entries.

Online Load Resume

Another On-line utility is introduced to prevent some more of those planned outages. On-line LOAD RESUME is available now effectively combining the speed and performance of the LOAD utility with the availability and access offered by INSERT processing. The Online LOAD (LOAD RESUME YES SHRLEVEL CHANGE) operates similar to an SQL INSERT program:

- It claims instead of draining
- Attempts to maintain the clustering order of the data
- It is LOG YES only (will not require a COPY afterward)

- Locking problems are avoided through internal monitoring of the commit scope
- Can also be run in parallel for partitioned tablespaces

Management Enhancements

Migration Options

You do have the ability to migrate (and fallback) from V5 or V6 to V7. Migration will be very fast and the CATMAINT process for V7 is done in three steps.

The first step will do all the mandatory catalog processing

The second step will look for any unsupported objects such as Type 2 indexes, and it will issue messages for these found objects

The third step will be optional for migrating stored procedures

For data sharing, you will have to have the fallback SPE PQ34467 installed for the migration. You can only have V5 and V7 existing together or V6 and V7, but not a combination of V5, V6 and V7.

Utility Lists, Parameters, and Dynamic Allocation

There is now support for utility wildcarding giving us the ability to execute utilities against a list of objects matching a specified pattern of matching characters. We will be able to execute a utility for a specified list of objects. There is a new LISTDEF statement that will also allow us to INCLUDE and/or EXCLUDE certain objects from these lists. Optionally you could create a utility procedure that would allow you to run a mixture of several utilities against several objects with one command. You will be able to use dynamic allocations instead of JCL. Any datasets required can also now be dynamically allocated for utilities, through the use of data set templates used to automatically generate datasets based upon a defined set of criteria.

```

TEMPLATE tmp1
  DSNAME (KBCE.&tsname..D&JDATE..COPY&ICTYPE.&PRIBAK.)
LISTDEF payroll
  INCLUDE TABLESPACE PAYROLL.*
  INCLUDE INDEXSPACE PAYROLL.*IX
  EXCLUDE TABLESPACE PAYROLL.TEMP*
COPY LIST payroll ... COPYDDN(tmp1,tmp1)

```

Statistics History

Enhancements made to elapsed time reporting and statistics collection. Now there is the ability to keep a history of statistics, which allows for better proactive performance analysis capabilities. It will help us better monitor our objects by providing the ability to monitor growth over time, and combined with other information, this will help us to determine if objects need to change.

This feature is supported via nine new catalog tables used to keep historical statistics. There is a new keyword of HISTORY in the RUNSTATS, REORG, LOAD and REBUILD utilities. Also, there is the ability to delete old statistics from the catalog history tables.

Statistics History – New Catalog Tables and New Options

```
SYSIBM.SYSCOLDIST_HIST  
SYSIBM.SYSCOLUMNS_HIST  
SYSIBM.SYSINDEXPART_HIST  
SYSIBM.SYSINDEXES_HIST  
SYSIBM.SYSINDEXSTATS_HIST  
SYSIBM.SYSLOBSTATS_HIST  
SYSIBM.SYSTABLEPART_HIST  
SYSIBM.SYSTABLES_HIST  
SYSIBM.SYSTABSTATS_HIST
```

```
REORG INDEX...HISTORY (ALL|ACCESSPATH|SPACE|NONE)  
RUNSTATS TABLESPACE...HISTORY (ALL|ACCESSPATH|SPACE|NONE)
```

New history option also on LOAD and REBUILD

Precompiler Services

The previous method of preparing an application program was to precompile in one job step, and then pass the output to another job step to invoke the compiler. Now there is a new alternative. This is called Precompiler Services and it will allow for a precompile and compile in one step. It provides for improved integration for host language support, more consistent rules, easier portability, and greater flexibility.

In the initial rollout however, this will only be available for COBOL.

Light Restart for Data Sharing

This will bring up just enough of a failed subsystem to release the retained locks and nothing more. Retained locks are released during DB2 restart when completed and in-commit transactions have their updates applied to disk. In-flight and in-abort transactions have their updates removed from disk. Following these updates, the locks can be safely released. DB2 can now be safely stopped and started again on another member of the sysplex group.

It is possible to complete 'cross system' restart faster by starting DB2 in Light mode. There is a reduced memory requirement, by reducing the number of service tasks. Only virtual buffer pools are used (max VPSIZE of 2000). The IRLM is started PC=YES to place the locks in private and save storage.

Control Center Enhancements

Control center has been enhanced to provide improved Utility Support for utility wildcarding. This is the ability to execute a utility for a specified list of objects along with restarting utilities from last committed phase (phase) or the last commit point (current) only for utilities originally started in the CC.

Control center now supports utility Ids, allowing for a utility ID template to be created using a variety of variables such as USERID and UTILNAME. In addition:

- Dataset Management -- Retrieve dataset lists, show, rename, and delete members
- DDL Generation -- Ability to recreate database objects and dependent objects, where the output can be saved to a dataset (maybe for input to SPUFI) or to a file
- Integrated SQL Assist -- Helps you build SQL statements and is available on Create Trigger window and Create View window

Index Advisor

This is a tool to assist you in choosing an optimal set of indexes for your table data. The initial rollout of the product will be done via an informal download from the IBM DB2 for OS/390 site. This reduces the need to design and define suitable indexes for your data. It does not do away with all index design strategies but will provide help for those finding the best indexes for a problem query or for a particular set of queries. It also helps testing an index on a workload w/out having to create the index. Index advisor will be able to work on a set of dynamic SQL statements, which have to be processed over a given period of time.

Only dynamic SQL will be supported in initial rollout (static coming later). It will use statistics from the system and user input to be able to evaluate strategies for given workloads and will also be able to scan the catalog tables holding SQL from bound applications to make determinations and suggestions. Prior to using, a modeling database must be created for DB2 for OS/390 on a DB2 for UNIX, Windows, OS/2 workstation using configuration parameters to mimic DB2 for OS/390.

DBADM Can Create Views For Others

DBADM now provides the ability to create view on behalf of others. This avoids the need to have to give SYSADM authority to a DBA. All other requirements of creating view still must be met. The view owner will have SELECT on the view without GRANT option, and the owner may drop the view if necessary. This feature will be controlled by the ZPARM DBACRVW.

e-business Enhancements

Improved DB2 Connect, JDBC and ODBC

As usual there are improvements in the DB2 Connect component. There are tracing and performance improvements with JDBC and ODBC, which are being heavily used in the e-business environment.

XML Support

An XML extender has been added to the DB2 extenders. This allows for a method of marking the meaning of the data for easier use. XML is an acronym for Extensible Markup Language. It is extensible in that the language itself is a meta-language that allows you to create your own language depending on the needs of your enterprise. It is possible to combine structured XML information with traditional relational data, and then choose whether to store

entire XML documents in DB2 as an XML Character Large Object (XMLCLOB), or map the XML content as traditional data in relational tables.

This also adds the ability for powerful searching of rich data types of XML element or attribute values, even for nontraditional XML data types. Incoming XML documents can be decomposed into traditional SQL data types and placed in columns. Traditional SQL data can also be used to compose outgoing XML documents. There are mapping methods implemented to provide for the transformation between XML documents and relational data. Facilities allow for decomposing an XML document into one or more pieces for storing in tables. Data in existing relational tables can be used to compose XML documents.

Kerberos Support

Kerberos security for Windows 2000 is supported in V7. Kerberos is a security technology to provide user and applications with secure access to resources anywhere in a heterogeneous network. It is a replacement for the standard DCE security. RACF will be implementing an interface to Kerberos, and in OS/390 Release 10 DCE support will be removed.

Conclusion

So where does that really leave us today, August 2000. If we follow the new style advice emanating from IBM, then we don't think about V7 until a minimum of 6 months after the GA release data, which will probably be in December of this year. If we are on Version 5 today, and are stable, and do not need the enhanced function of V6, then stay there and move to V7 when the new functions are needed. V5 is the stable production grade release at this moment. I am sure V6 will become that stable, but not for some time yet, due again (in my opinion) to all the retrofits, and the fixes to the retrofits and the problems the retrofits caused.

I would much prefer to see IBM deliver maintenance releases more frequently, but with more testing like the QPP program. The V6 retrofits did not go through a QPP and yet they are rather extensive – again in my opinion.

But suppose you are V6 today, and appear to be working just fine, what should be your strategy? Just stay on V6, current as possible on maintenance. If you truly need the V7 enhancements, then get in the QPP program for V7, help IBM work out the problems, and then be ready with it when it comes of age.

As far as the issue of utilities and pricing and unbundling...if the function and speed required by you is satisfied by the IBM DB2 utilities, use them, or come back to them. If they do not meet your requirements, then use the vendor's utilities on a utility set by utility set basis. This also has to do with you time of migration to new releases. If you are bleeding edge, then stay with IBM. If you stay off the new releases for at least a year, then you have more freedom in utilities, but less freedom in function. I fully believe that IBM will make strong inroads into becoming a utility vendor. Does that mean that there is a database vendor on the market that will deliver something competitive to DB2? Not likely in the near future but as with any technology, if IBM becomes stagnant due to its success, then it will open the door again to competition as it did in the past.

Richard is an internationally recognized consultant, lecturer and teacher, known for his expertise in enterprise information systems. Richard is widely published and has written columns for several magazines. He is a regular

guest speaker at conventions and user groups in the U.S., Australia, Japan and Europe, and has won best-speaker awards for many of these. He is one of the authors of "Data Warehouse: Practical Advice from the Experts", published by Prentice-Hall, "DB2 Answers", published by Osborne-McGraw-Hill, "DB2 High Performance Design and Tuning", published by Prentice-Hall, and to be released in 2001 "DB2 SQL: Design, Optimization, Performance, Tuning", for the entire DB2 family, published by Prentice-Hall. Formerly a senior consultant with Codd and Date Inc., Richard is currently a principal with YL&A, a firm specializing in database performance, technical research, and advanced education. Richard is a member of IBM's DB2 and S/390 Sysplex Gold Consultants programs. You can reach him via email at Richard_Yevich@YLAssoc.com or via telephone at 1-888-246-5049.