IN THIS APPENDIX

- E-Business Enhancements
- Application Development Enhancements
- Data Management Enhancements
- Business Intelligence Enhancements
- Additional V7 Information

APPENDIX **F**

DB2 Version 7 Overview

This appendix contains a short overview of the highlights of DB2 Version 7. Refer to this appendix when you are interested in finding out about specific features that were added as of DB2 V7.

IBM officially released DB2 V7 for the mainframe at the end of March 2001. Keep in mind, too, that IBM enabled support for some of the features of V7 via APAR to V6 and via a refresh of the V6 code prior to V7 availability. For more information on what APARs are available for V6, consult the following Web link:

```
http://www.software.ibm.com/data/db2/os390/v6apar.html
```

This appendix breaks down the new features of this release into the following categories:

- e-Business
- Application
- Data Management
- Business Intelligence
- Additional Considerations

E-Business Enhancements

The Internet is pervasive. It affects almost every aspect of our IT infrastructure. Almost every business today is developing more Web-enabled applications. In other words, businesses are transforming themselves into e-businesses.

DB2 V7 provides assistance to e-businesses by supporting XML. XML stands for eXtensible Markup Language. XML allows tags to be defined by users that describe the data in the document. This helps to make the document somewhat self-describing. XML is quickly becoming the de facto standard for application interfaces, as well as for inter- and intra-organization data transfer.

DB2 V7 supports XML using a new data type extender for XML documents: the XML Extender. The XML Extender is similar to the other extenders for video, image, audio, and text that were added to DB2 V6. The DB2 Extenders combine user-defined distinct types, user-defined functions, and triggers to provide extended data type functionality for DB2 databases.

The XML Extender enables XML documents to be integrated with DB2 databases. By integrating XML into DB2 databases, you can more directly and quickly access the XML documents. You can search and store entire XML documents using SQL. You also have the option of combining XML documents with traditional data stored in relational tables.

When you store or compose a document, you can invoke DBMS functions to trigger an event to automate the interchange of data between applications. An XML document can be stored complete in a single text column, or XML documents can be broken into component pieces and stored as multiple columns across multiple tables.

The XML Extender provides user-defined data types (UDTs) and user-defined functions (UDFs) to store and manipulate XML in the DB2 database. The XML Extender defines UDTs for `XMLVARCHAR`, `XMLCLOB`, and `XMLFILE`. After the XML is stored in the database, the UDFs can be used to search and retrieve the XML data as a complete document or in pieces. The UDFs supplied by the XML Extender include

- **Storage** functions to insert XML documents into a DB2 database
- **Retrieval** functions to access XML documents from XML columns
- **Extraction** functions to extract and convert the element content or attribute values from an XML document to the data type that is specified by the function name
- **Update** functions to modify element contents or attribute values (and to return a copy of an XML document with an updated value)

Additionally on the e-business front, Net.Data has been enhanced to provide built-in XML exploitation. You can generate XML tags as output from Net.Data macros and use XML style sheets to format and display the generated output.

Application Development Enhancements

The second category of enhancements pertains to application development and programming. DB2 V7 offers many new features to simplify the process of programming DB2 applications, thereby helping developers become more productive.

Stored Procedure Enhancements

Stored Procedure Builder (SPB) is a new feature that provides a point-and-click environment for building stored procedures. The SPB can be used to develop stored procedures for both the distributed and mainframe DB2 environments. The SPB can be used either standalone or in conjunction with a development tool (such as IBM VisualAge, Microsoft Visual Basic, and Microsoft Visual Studio). SPB supports SQL Procedure Language and Java as stored procedure host languages.

The second big application enhancement is SQL Procedure Language support. SQL Procedure Language enables stored procedures to be written in an extended, procedural SQL language. IBM's SQL Procedure Language is compatible with the ANSI SQL/PSM specification. It extends the SQL language to support additional functionality, effectively making SQL a more computationally complete language. Examples of the extended programming capabilities added to SQL for SQL Procedure Language include

- Assignment statements
- `CASE - LEAVE`
- Cursors
- `IF - THEN - ELSE`
- Local variables
- `LOOP`, `REPEAT`, and `WHILE`
- `FOR`, `CALL`, and `RETURN`
- `GET DIAGNOSTICS`
- `SIGNAL` and `RESIGNAL`

Before SQL Procedure Language programs can be executed, first the code needs to be converted into C by the Stored Procedure Builder. Once converted, the code goes through the program preparation process. So, you get the benefit of writing code using the simple SQL Procedure Language dialect and the performance benefit of optimized C code. However, you will need to own a C compiler to take advantage of SQL Procedure Language.

Finally, for stored procedures, as of V7 DB2 provides the capability to issue `COMMIT` and `ROLLBACK` statements inside a stored procedure. The `COMMIT` or `ROLLBACK` will affect the entire unit of work, including any work done by the calling program, not just the work done within the stored procedure itself. So, you will need to use caution when issuing a `COMMIT` or `ROLLBACK` within a stored procedure.

Scrollable Cursors

Probably the most significant new application development enhancement made to DB2 for V7 is scrollable cursors. A scrollable cursor provides the ability to scroll forward and backward through the data once the cursor is open. This can be achieved using nothing but SQL—no host language code (COBOL, C, and so on) is required to facilitate a scrollable cursor in DB2 V7. A scrollable cursor makes navigating through SQL result sets much easier. There are two types of DB2 scrollable cursors: `SENSITIVE` and `INSENSITIVE`.

A `SENSITIVE` scrollable cursors is updateable, meaning it can access data changed by the user or other users. An `INSENSITIVE` scrollable cursor, however, is not updateable, so it will not show any changes made.

To use scrollable cursors, you must use declared temporary tables, another new feature of DB2 Version 7. Declared temporary tables are discussed later in this appendix in the section "Data Management Enhancements." DB2 uses a declared temporary table to hold and maintain the data returned by a scrollable cursor.

Scrollable cursors allow developers to move through the results of a query in multiple ways. The following keywords are supported when fetching data from a scrollable cursor:

- `NEXT`—Will `FETCH` the next row, the same way that the pre-V7 `FETCH` statement functioned
- `PRIOR`—Will `FETCH` the previous row
- `FIRST`—Will `FETCH` the first row in the results set
- `LAST`—Will `FETCH` the last row in the results set
- `CURRENT`—Will re-`FETCH` the current row from the result set
- `BEFORE`—Positions the cursor before the first row of the results set
- `AFTER`—Positions the cursor after the last row of the results set
- `ABSOLUTE` *n*—Will `FETCH` the row that is *n* rows away from the first row in the results set
- `RELATIVE` *n*—Will `FETCH` the row that is *n* rows away from the last row fetched

For both `ABSOLUTE` and `RELATIVE`, the number *n* must be an integer. It can be either a positive or a negative number, and it can be represented as a numeric constant or as a host variable.

All of the `FETCH` options for scrollable cursors also reposition the cursor before fetching the data. For example, consider the following cursor logic:

```
DECLARE csr1 SENSITIVE STATIC SCROLL CURSOR
FOR SELECT   FIRSTNAME, LASTNME
   FROM     DSN8710.EMP
   ORDER BY LASTNME;

OPEN csr1;

FETCH LAST csr1 INTO :FN, :LN;
```

Issuing this SQL will declare a scrollable cursor named `csr1`, open that cursor, and then `FETCH` the last row from the cursor's results set. The `FETCH LAST` statement will reposition the cursor to the last row of the results set, and then `FETCH` the results into the host variables as specified. Scrollable cursors reduce the amount of time and effort required to move backward and forward through the results of SQL queries.

But as helpful as scrollable cursors are, do not make every cursor a scrollable cursor. Scrollable cursors require substantially more overhead than a traditional, non-scrollable cursor. Analyze the requirements of your applications and deploy scrollable cursors only where it makes sense to do so.

Limiting the Number of Rows Fetched

Application developers frequently need to retrieve a limited number of qualifying rows from a table. For example, maybe you need to list the top ten best selling items from inventory. There are several ways to accomplish this prior to DB2 V7 using SQL, but they are not necessarily efficient.

The first reaction is to simply use the `WHERE` clause to eliminate non-qualifying rows. But this is simplistic, and often is not sufficient to produce the results desired in an optimal manner. What if the program only requires that the top ten results be returned? This can be a somewhat difficult request to formulate using SQL alone. Consider, for example, an application that needs to retrieve only the ten most highly paid employees from the `EMP` sample table. You could simply issue a SQL request that retrieves all of the employees in order by salary, but only use the first ten retrieved. That is easy, for example

```
SELECT   EMPNO, FIRSTNME, LASTNAME, SALARY
FROM     DSN8710.EMP
ORDER BY SALARY DESC;
```

You must specify the `ORDER BY` clause with the `DESC` keyword. This sorts the results into descending order, instead of the default, which is ascending. Without the `DESC` keyword, the "top ten" would be at the very end of the results set, not at the beginning.

But that does not really satisfy the requirement—retrieving only the top ten. It merely sorts the results into descending sequence. So, the results would still be all employees in the table, but in the correct order so you can view the "top ten" salaries very easily. The ideal solution should return only the ten employees with the highest salary and not merely a sorted list of all employees.

You can code some "tricky" SQL to support this request for all versions of DB2, such as the following:

```
SELECT   EMPNO, FIRSTNME, LASTNAME, SALARY
FROM     DSN8710.EMP A
WHERE 10 > (SELECT COUNT(*)
            FROM   DSN8710.EMP B
            WHERE A.SALARY < B.SALARY
            AND   B.SALARY IS NOT NULL)
ORDER BY SALARY DESC;
SELECT   EMPNO, FIRSTNME, LASTNAME, SALARY
FROM     DSN8710.EMP A
WHERE 10 > (SELECT COUNT(*)
            FROM   DSN8710.EMP A
            WHERE A.SALARY < B.SALARY)
AND SALARY IS NOT NULL
ORDER BY SALARY DESC;
---------+---------+---------+---------+---------+---------+---------+-
DSNT408I SQLCODE = -206, ERROR:  B.SALARY IS NOT A COLUMN OF AN INSERTE
         UPDATED TABLE, OR ANY TABLE IDENTIFIED IN A FROM CLAUSE, OR IS
         COLUMN OF THE TRIGGERING TABLE OF A TRIGGER
```

This SQL is portable from version to version of DB2 (as well as to another DBMS, such as Oracle or SQL Server). And, of course, you can change the constant 10 to any number you

wish, thereby retrieving the top 20, or top 5, as deemed necessary by the needs of your application. Because the SALARY column is nullable in the EMP table, you must remove the nulls from the results set. The ORDER BY is required to sort the results in the right order. If it is removed from the query, the results will still contain the top ten, but they will be in no particular order.

DB2 V7 provides an easier and less complicated way to limit the results of a SELECT statement—the FIRST keyword. You can code FETCH FIRST *n* ROWS, which will limit the number of rows that are fetched and returned by a SELECT statement. Additionally, you can specify a new clause—FETCH FIRST 1 ROW ONLY—on SELECT INTO statements when the query can return more than one row in the answer set. Doing so informs DB2 to ignore any other rows.

There is one difference between the new V7 formulation and the other SELECT statement we reviewed, and that is the way "ties" are handled. A tie occurs when more than one row contains the same value. The previous query we examined might return more than 10 rows if there are multiple rows with the same value for price within the top ten. Using the FIRST keyword, DB2 will limit the number of rows returned to ten, even if there are other rows with the same value for price as the number ten row in the results set. The needs of your application will dictate whether ties are to be ignored or included in the result set. If all "ties" need to be included in the results set, the new V7 feature might not prove to be helpful.

External SAVEPOINTS

DB2 V7 allows you to set a SAVEPOINT within a transaction. You can think of a SAVEPOINT as a sub-UOW (unit of work) "stability" point. You can code application logic to undo any data modifications and database schema changes that were made since the application set the SAVEPOINT. Application development should be more efficient using SAVEPOINTs because you will not need to include contingency and what-if logic in your application code.

Issuing a SAVEPOINT does not COMMIT work to DB2. It is simply a mechanism for registering milestones within a transaction or program. Let's learn by example. Consider the following pseudo-code:

```
SAVEPOINT POINTX ON ROLLBACK RETAIN CURSORS;

Subsequent processing. . .

ROLLBACK TO SAVEPOINT POINTX;
```

The ROLLBACK will cause any data or schema changes made in the "subsequent processing" to be undone.

It is permissible to code multiple SAVEPOINTs within a UOW, and you can ROLLBACK to any SAVEPOINT (as long as you do not reuse the SAVEPOINT name). The UNIQUE keyword can be specified to ensure that the SAVEPOINT name is not reused within the unit of recovery.

There are two clauses that can be specified to further define the nature of the SAVEPOINT when a ROLLBACK is issued:

- RETAIN CURSORS—Specifies that any cursors opened after the SAVEPOINT is set are not tracked and will not be closed when rolling back to that SAVEPOINT.
- RETAIN LOCKS—Specifies that any locks acquired after the SAVEPOINT is set are not tracked and will not be released when rolling back to the SAVEPOINT.

Even if RETAIN CURSORS is specified, some of the cursors might not be useable. For example, if the ROLLBACK removes a row (that is, rolls back an INSERT) upon which the cursor was positioned, an error will arise.

Row Expressions

SQL becomes even more flexible under DB2 V7 with row expressions. Row expressions allow SQL statements to be coded using more than one set of comparisons in a single predicate using a subquery. The net result is that multiple columns can be compared within the scope of a single SQL predicate—possibly against multiple rows on the right side of the predicate. Once again, the best way to understand this feature is by viewing an example:

```
SELECT *
FROM   SAMPLE_TABLE
WHERE  (COL1, COL2) IN (SELECT COLX, COLY
                        FROM   OTHER_TABLE);
```

You can readily see the difference: Two columns are coded on the left side of the predicate, thereby enabling two columns to be selected in the SELECT statement on the right side of the predicate. Of course, a row expression need not be limited to only two columns; multiple columns can be specified, so long as the number of columns on the left matches the number of columns on the right side of the predicate. Row expressions bring more flexibility and can greatly simplify certain types of SQL statements.

SQL Assist

Another feature that will aid application developers is SQL Assist. The SQL Assist feature is a GUI-driven tool to help you build SQL statements such as SELECT, INSERT, UPDATE, and DELETE. It is accessible from the following "products":

- Control Center
- Stored Procedure Builder
- Data Warehouse Center

Precompiler Services

DB2 V7 supports more robust Precompiler Services. An API is provided that can be called by a host language compiler or preprocessor. Precompiler Services enable developers to precompile and compile programs in a single step, instead of multiple steps. This makes the development environment more flexible, easier to use, and able to offer better

portability between members of the DB2 Family. Initial support for Precompiler Services is provided for COBOL only, but support for other languages is planned for later DB2 releases.

Additional Application Development Improvements

IBM has made numerous additional improvements to application development aspects of DB2 in Version 7. Some of the more interesting enhancements include

- Improving DB2's support of JDBC and ODBC, including support for JDBC 2.0 and ODBC 3.0.
- Improvements in SQL optimization and better parallel query support.
- The ability to run ODBC/CLI programs as a static application (instead of only as dynamic).
- Support for encouraging or discouraging index access for small tables. A DSNZPARM value is provided that can be set to give the DB2 optimizer guidance on the threshold for what constitutes a small table in your shop.
- The ability to code a self-referencing sub-SELECT on searched UPDATE and DELETE statements. In previous releases of DB2, the WHERE clause cannot refer to the table (or view) being modified by the statement. For example, the following SQL is legitimate as of DB2 V7 and can be used to implement a 10% raise for employees who earn less than their department's average salary:

```
UPDATE DSN8710.EMP E1
SET SALARY = SALARY * 1.10
WHERE SALARY < (SELECT AVG(SALARY)
                FROM   DSN8710.EMP E2
                WHERE  E1.WORKDEPT = E2.WORKDEPT);
```

  DB2 will evaluate the complete subquery before performing the requested UPDATE.

Data Management Enhancements

The third grouping of DB2 V7 enhancements addresses data management and database administration issues. DB2 V7 offers extended capabilities for managing data more effectively and helping DBAs to be more productive.

Identity Columns

A common requirement of relational applications and databases is the need to store a counter that identifies rows in tables. Until V7, DB2 provided no inherent support for such functionality. DB2 V7 adds support for IDENTITY columns.

An IDENTITY column can be defined to a DB2 table such that DB2 will automatically generate a unique, sequential value for that column when a row is added to the table. For example, IDENTITY columns can be used to generate unique primary key values. DB2's implementation of IDENTITY columns avoids some of the concurrency and performance problems that can occur when application programs are used to populate sequential values for a "counter" column.

When inserting data into a table that uses an IDENTITY column, the developer or user does not provide a value to be inserted for the IDENTITY column. Instead, DB2 will calculate the appropriate value to be inserted.

Only one IDENTITY column can be defined per DB2 table. Additionally, the data type of the column must be SMALLINT, INTEGER, or DECIMAL with a zero scale, that is DECIMAL(x,0). The data type also can be a user-defined DISTINCT type based on one of these numeric data types. The designer has control over the starting point for the generated sequential values and the number by which the count is incremented.

The following example creates a table with an IDENTITY column:

```
CREATE TABLE EXAMPLE
   (ID_COL INTEGER NOT NULL
           GENERATED ALWAYS AS IDENTITY
           START WITH 100
           INCREMENT BY 10
    ...);
```

In this example, the IDENTITY column is named ID_COL. The first value stored in the column will be 100, and subsequent INSERTs will add 10 to the last value. The identity column values generated will be 100, 110, 120, 130, and so on.

Declared Temporary Tables

Declared temporary tables complement the existing DB2 (V5) capability to create global temporary tables, but declared temporary tables differ from global temporary tables in many significant ways:

- Declared temporary tables do not have descriptions in the DB2 Catalog. They are defined in the program, instead of prior to program execution.
- Declared temporary tables can have indexes and CHECK constraints defined on them.
- You can issue UPDATE statements and positioned DELETE statements against a declared temporary table.
- You can implicitly define the columns of a declared temporary table and use the result table from a SELECT.

Declared temporary tables are much more functional than global temporary tables. An instance of a declared temporary table can be created using the DECLARE GLOBAL TEMPORARY TABLE statement. That instance of the table is known only to the process that issues the DECLARE statement. Multiple concurrent programs can be executing using the same declared temporary table name because each program will have its own copy of the declared temporary table.

Before you can declare temporary tables, you must create a temporary database and table spaces for them to use. This is accomplished by specifying the AS TEMP clause on a CREATE DATABASE statement. Then, you must create segmented table spaces in the temporary database. Only one temporary database for declared temporary tables is permitted per DB2 subsystem.

When a DECLARE GLOBAL TEMPORARY TABLE statement is issued, DB2 will create an empty instance of the temporary table in the temporary table space. INSERT statements are used to populate the temporary table. Once inserted, the data can be accessed, modified, or deleted. When the program completes, DB2 will drop the instance of the temporary table.

The following example shows a DECLARE statement that can be issued from an application program (assuming the temporary database and table spaces have been defined):

```
DECLARE GLOBAL TEMPORARY TABLE TEMP_EMP
  (EMPNO      CHAR(6)     NOT NULL,
   FIRSTNME   VARCHAR(12) NOT NULL,
   MIDINIT    CHAR(1)     NOT NULL,
   LASTNAME   VARCHAR(15) NOT NULL,
   WORKDEPT   CHAR(3),
   PHONENO    CHAR(4)
  );
```


Additionally, you can use the LIKE clause to DECLARE a temporary table that uses the same schema definition as another currently existing table. You can use the INCLUDING IDENTITY COLUMN ATTRIBUTES clause to copy the IDENTITY columns as well.

SQL statements that use declared temporary tables may run faster because DB2 limits the amount of logging and locking performed. Declared temporary tables can be useful in the following scenarios as well:

- When you need to retrieve data once and use it repetitively throughout a program, especially if the cost to retrieve the data is high (because the cost of retrieving it from a declared temporary table may be lower).
- When you wish to retrieve data from non-relational data sources (flat file, IMS, IDMS, and so on) and use SQL to access it or join it to other DB2 data.

It actually might be more appropriate to classify declared temporary tables as an application development enhancement, because they must be defined (declared) and used within the context of an application program. However, since they act like tables—a database object—declared temporary table support is grouped under Data Management enhancements.

Unicode Support

Support for an additional encoding scheme, Unicode, is added for DB2 V7. Unicode can be specified as the default on the DEF ENCODING SCHEME parameter of the DSNTIPF installation panel or when creating databases and table spaces using the CCSID parameter.

Unicode is an encoding scheme, like ASCII or EBCDIC, but it is more than that. This is so because Unicode provides a unique number for every character, no matter what the platform, program, or language. Unicode is required by modern computing standards, such as XML, Java, LDAP, and CORBA 3.0, WML, and is the official way to implement ISO/IEC 10646. The emergence of the Unicode Standard is significant in furthering a truly global computing and software environment.

Unicode is useful for multinational support, because it can be used to represent characters of virtually all languages. More information about Unicode can be found at `http://www.unicode.org`.

Utility Improvements

As is the case for each new release of DB2, IBM has provided numerous enhancements to the functionality and speed of the DB2 utilities. Some of the more interesting enhancements include:

- `UNLOAD` utility—A true utility that provides better speed than `DSNTIAUL`. `DSNTIAUL`, until recently the primary method of unloading data from DB2 tables, is a sample program, not a true utility. As such, it lagged in features and functionality and many organizations chose to purchase unload capability from third-party ISVs like BMC Software and CDB Software. The new IBM `UNLOAD` utility though, is not free, but must be purchased from IBM as part of a DB2 utilities package at an extra charge.
- Parallel `LOAD`—V7 provides the capability to load a partitioned table space with multiple input data sets in a single step.
- Online `LOAD RESUME`—V7 provides the ability to add data to a table while the data in the table remains available.
- `COPYTOCOPY` utility—This is a new utility that creates additional, registered image copies from existing image copies. The input or output can be local primary, local backup, offsite primary, or offsite backup copies. The generated copies can be used just like any other DB2 image copy backup.
- Speed—as with each new release of DB2, IBM claims that each utility will run faster than earlier versions of the utilities.

However, the most important new development with IBM's utilities is automatic data set allocation and the ability to specify lists of objects with wildcarding. Utilities from the third-party ISVs have offered similar capabilities for several years now, and many DB2 users have been clamoring for IBM to provide similar functionality.

With automatic data set allocation, the utilities determine which data sets are required to perform the function and what size the data sets need to be. This helps, because each utility requires different data sets as they operate to save data during interim steps. Automatic allocation saves the DBA the effort of determining the information before running the utility. Additionally, "out of space" errors (for example, `SB37`) can be avoided because the size of the work data sets will be determined prior to each utility run, and need not be recalculated manually as database objects increase in size.

DB2 V7 provides the ability to create templates for utility data sets. Specifying templates to the dynamic data set allocation process provides needed data set characteristics. Both `DASD` and `TAPE` templates can be specified. Options are available to support features such as GDG generation and tape stacking.

Another new utility feature is the ability to supply lists of objects to a utility for processing. The `LISTDEF` parameter can be used to create these lists of objects for utility processing. The

LISTDEF specification can use wildcarding to rapidly specify multiple objects without having to explicitly name each of the objects. For example, you can specify

```
LISTDEF DB1 INCLUDE TABLESPACE DBXN.*
            EXCLUDE TABLESPACE DBXN.TS2

REORG LIST DB1 . . .
```

This will reorganize all table spaces in the database named DBXN except for the one table space exempted, namely TS2. Furthermore, if a table space is subsequently added to DBXN, the REORG job does not need to be changed. The next time it runs, REORG will query the DB2 Catalog to determine the table spaces that exist in the list name DB1. Because it specifies all table spaces in DBXN, any new table space added to DBXN will automatically be picked up for processing.

The LISTDEF capability is very powerful. The LISTDEF definition can be specified either in a separate data set or in the SYSIN data set preceding a utility control statement. The default DD name for a LISTDEF statement is SYSLISTD.

DB2 provides multiple wildcarding options for the LISTDEF specification. The developers tried to support both de facto wildcarding standards, such as the asterisk (*), as well as the wildcarding options used by the SQL LIKE predicate. Pattern-matching characters available for wildcarding include

- Both the percent sign character (%) and the asterisk character (*) to represent zero or more characters
- The question mark character (?) to represent any single character

There are limits to the LISTDEF clause though. You cannot specify all-inclusive lists, such as DATABASE * or TABLESPACE *.*. But there are other powerful options such as the RI parameter that will include all tables referentially connected to the table(s) specified in the list. List generation and wildcarding can greatly simplify the creation and management of DB2 utility jobs.

And very importantly, with DB2 V7 IBM no longer ships the DB2 utilities free of charge with a DB2 license. This issue is discussed later in this appendix.

Deferred Data Set Creation

One of IBM's largest recent objectives has been to add features to DB2 to support ERP packages, such as SAP R/3 and Peoplesoft. Deferred data set creation, new as of V7, is one such feature. With deferred data set definition it is possible to create tables and not define the underlying data sets. Creation of the data sets to store data for the tables is deferred until they are used.

This is important for ERP packages where many tables are defined but never used. ERP packages are typically broken up into multiple business functions, and the customer can buy the software by functionality. But many ERP vendors simply create all of the database objects for all of the functionality of the entire package, regardless of the functionality purchased by the user. This causes many database objects to be defined but never used.

With deferred data set creation, customers deploying ERP packages on DB2 for OS/390 and z/OS can defer the physical creation of the underlying data sets for the database objects, while allowing the ERP package to create all of the database objects it requires.

Log Suspend/Resume

DB2 V7 provides better support for copying data at the storage hardware level. The new `LOG SUSPEND` command can be used to halt `UPDATE` activity and logging. Additionally, the `LOG RESUME` command is used to restart update activity and logging.

The log suspension and resumption commands make it easier to make external copies of the system. After issuing `LOG SUSPEND`, you can use a fast-disk copy facility, such as FlashCopy on IBM's Shark ESS or SnapShot on a RAMAC Virtual Array. After the fast snap is completed, `LOG RESUME` can be issued to re-enable database modification.

Data Sharing Enhancements

DB2 V7 provides several improvements for data-sharing environments. One such enhancement is referred to as ***Restart Light***. To support this new light restart option, the `START` DB2 command has been enhanced. Restart Light allows a DB2 data-sharing member to restart with a minimal storage footprint and then to terminate normally after DB2 frees retained locks. By reducing storage requirements, restart for recovery can be possible for more resource-constrained systems.

Improved immediate write capability is another data sharing enhancement. DB2 V6 provided an option to immediately write updated group buffer pool dependent buffers. DB2 V7 enhances this capability by recording the choice in the DB2 Catalog and externalizing it on the installation panels.

A final V7 data sharing improvement is support for persistent structure size changes. As of DB2 V7, changes made to structure sizes using the `SETXCF START, ALTER` command will be persistent when you rebuild or reallocate a structure.

Additional Data Management Enhancements

IBM has made numerous additional improvements to the data management capabilities of DB2 in Version 7. Some of the more interesting enhancements include

- The ability to change most `DSNZPARM`s without first stopping and then restarting DB2.
- Support for coding `UNION` and `UNION ALL` in views. This support is added not just for `CREATE VIEW`, but also for inline views (where a `SELECT` statement is coded in the `FROM` clause of another `SELECT` statement).
- Instead of `RUNSTATS` always obliterating any old statistic values, a history of object statistics can be maintained in the DB2 Catalog. This way DBAs can review the historical growth and changes for database objects. Support has been added for the `MODIFY STATISTICS` to be able to remove historical statistics from the DB2 Catalog.
- The DB2 Control Center has been enhanced. One of the biggest enhancements is the ability to generate DDL from the DB2 Catalog using DB2 Control Center.

- Users with DBADM authority can create views for others, thereby minimizing the reasons for granting SYSADM.
- Sometime after V7 general availability (but before V8) IBM released real-time statistics. This feature brings to DB2 the ability to capture performance statistics and populate them into catalog-like tables, as DB2 runs—without the need to run an external utility.
- The ability to issue DDF SUSPEND and DDF RESUME commands to temporarily halt activity from requesters without terminating connections. The primary reason to suspend DDF requests is to enable DDL statements issued on the server to complete.

Business Intelligence Enhancements

The final broad category of DB2 V7 enhancements is to better enable creation, management, and access of DB2 data warehouses built on the mainframe. DB2 V7 supports a new management tool called the Data Warehouse Manager.

The DB2 Data Warehouse Manager makes it easier to use DB2 for data warehousing and business intelligence applications. It is integrated with DB2 Control Center. The predominant capabilities provided by DB2 Data Warehouse Manager include

- The ability to control and govern data warehouse queries
- Help for data cleansing, generating key columns, generating period tables, and inverting and pivoting tables
- Statistical transformers for business intelligence operations, such as subtotals, rollups, cubes, moving averages, regression, and so on
- Data replication to allow heterogeneous data movement between data warehouse sources and targets

The DB2 Data Warehouse Manager will make it significantly easier for technicians to deploy useful data warehouses using DB2 as the data store.

Additional V7 Information

A simple discussion of the new features and enhancements IBM has made to DB2 for V7 actually provides incomplete coverage of Version 7. There are several nuances of the new release that also must be discussed.

The first issue is IBM's new utility packaging. As of V7, only a subset of base utilities will ship for free with DB2. Customers will now have to purchase the IBM DB2 utilities as a package (at an additional cost). In every past release of DB2, customers received the utilities (such as LOAD, RECOVER, and REORG) at no charge as part of their DB2 software package.

Most customers will likely choose to purchase the full suite of IBM DB2 utilities, even if they use ISV utilities to enhance performance and functionality. This is the case because of how IBM chose to package the utilities. IBM offers three utility packages:

- **Operational Utilities**—COPY, EXEC, LOAD, REBUILD, RECOVER, REORG, RUNSTATS, STOSPACE, and UNLOAD
- **Recover & Diagnostic Utilities**—CHECK, COPY, COPYTOCOPY, MERGE, MODIFY RECOVERY, MODIFY STATISTICS, REBUILD, RECOVER
- **Utilities Suite**—Both Operational Utilities and Recover & Diagnostic Utilities as a single package

Organizations will need to evaluate the IBM DB2 utilities along with ISV DB2 utilities to determine what utilities are needed at their site. And, of course, you will need to consider the budget impact of purchasing utilities, because they will now be a separate line item on your purchase orders.

CAUTION

Proceed with caution before deciding to eliminate the IBM DB2 utilities. Some third-party utilities call the IBM utilities for certain functionality. Additionally, some of the IBM utilities use interfaces to DB2 that are not documented for third party use. Be sure to consult with your vendors to understand the various nuances of utility functionality and performance before choosing your DB2 utility vendor(s).

The second issue is the formal announcement made by IBM in late 2000 that they officially entered the DBA tools business. IBM announced several DBA tools for performance management, recovery management, application development, and database administration that will compete with the more entrenched DBA tools vendors (such as BMC Software and Computer Associates).

Do not misunderstand IBM's intent here. They plan to sell these DBA tools—they are not giving them away with DB2. So users do not get these additional DBA tools simply by migrating to DB2 Version 7. They will need to buy the tools from IBM the way they do today from other DBA tool ISVs. Also, keep in mind that IBM has provided some DBA tools for a long time (such as DB2-PM), so not all of these tools are new. And the tools are not all developed by IBM; some are simply DBA tools from other ISVs that are marketed and sold by IBM.

One final issue: migration. It will be possible to migrate to V7 directly from V5 without first migrating to V6. However, before deciding to make such a dramatic migration, please take time to consider the impact on your organization. DB2 V6 was a very large release of DB2. Additionally, DB2 V6 removed features from DB2 that were supported for several releases. Refer to Appendix H, "Reorganizing the DB2 Catalog," for exhaustive coverage of the features removed from DB2 as of V6.

Version 6 was the first release of DB2 to remove features. So, you will need to not only prepare for all the new features added to DB2 V7, but also all the new features added to DB2 V6, and ensure that you are not using any of the features removed from DB2 V6. If you are on V5 and want to move quickly to V7 it might be better to plan a staged migration where you move to V6 first, and only after several weeks of V6 operation plan to move to V7. Just because it is possible to move directly from V5 to V7 does not necessarily make it a wise idea.