

WEB 1

Special Consideration for DB2 Version 6

DB2 Version 6 was officially released in June 1999. This release included many new features such as large objects, triggers, user-defined functions, user-defined data types, and more. However, the item that is most important for every DB2 shop to understand is what is *not* in DB2 V6. For the first time, IBM removed features from DB2. As such, your organization must prepare its DB2 subsystems for V6 by removing the soon-to-be-unsupported features from your installation.

At this late date, this information is included here for those few shops that still need to migrate to Version 6, as well as for posterity for folks investigating obsolete DB2 features.

Let's examine each of the features that were removed from DB2 as of Version 6.

Type 1 Indexes

Prior to DB2 Version 6 there were two types of indexes available to DB2: type 1 and type 2. Type 2 indexes were introduced with DB2 Version 4 and are now the standard index type implemented in every DB2 shop. Even prior to V6, most organizations favored creating type 2 indexes over type 1 indexes because they offer the following benefits:

- Eliminate index locking (the predominant cause of contention in most pre-V4 DB2 applications).
- Type 2 indexes do not use index subpages.
- Type 2 indexes are the only type supported for ASCII encoded tables.
- Many newer DB2 features cannot be used unless Type 2 indexes are used; these features include row level locking, data sharing, full partition independence, uncommitted reads (ISOLATION(UR)), UNIQUE WHERE NOT NULL, and CPU and Sysplex parallelism.

IN THIS CHAPTER

- Type 1 Indexes
- Shared Read Only Data
- RECOVER INDEX
- Host Variables Without Colons
- Dataset Passwords
- Stored Procedure Registration
- Synopsis

As of DB2 V6, type 1 indexes are no longer supported by DB2. So, if you are running V6, V7, or V8, all of your shop's indexes will be type 2.

For those few shops not yet on Version 6, it is wise to begin migrating from type 1 to type 2 indexes as soon as possible, not just because of the benefits outlined earlier, but also because type 1 indexes are obsolete.

NOTE

If your shop is running DB2 V3 or an earlier release, you cannot implement type 2 indexes because they are not supported. If you are running on such an old version of DB2, you should really begin to migrate to a more recent DB2 version.

DB2 V4 was the first version of DB2 to begin supporting type 2 indexes.

To find all type 1 indexes in your DB2 subsystems issue the following SQL statement:

```
SELECT CREATOR, NAME
FROM SYSIBM.SYSINDEXES
WHERE INDEXTYPE = '1';
```

For DB2 V4 and V5 subsystems type 1 indexes are still supported. However, you should convert to type 2 indexes as soon as possible because of the benefits they provide. Additionally, you can set the DSNZPARM parameter DEFIXTP=2 to make type 2 indexes the default index type.

Shared Read Only Data

Shared read only data (SROD) was provided as a new feature of DB2 in Version 2.3. SROD provided a way for the same DB2 database to be read by multiple DB2 subsystems without implementing distributed data or Sysplex data sharing. However, the shared object must be started ACCESS(RO), and all data access is read only. When the data needs to be updated, only one of the subsystems, the one marked as the owner, can update the data.

SROD is complex to implement, limited in functionality, and not frequently implemented. Subsequent functionality, such as data sharing and more functional distributed data support, has supplanted the need for SROD capability. As of DB2 V6, SROD support is removed. To support SROD-like functionality, you will need to convert to using distributed DB2 databases or data sharing.

To find all databases defined as shared read only, execute the following SQL statement:

```
SELECT NAME, BPOOL, ROSHARE
FROM SYSIBM.SYSDATABASE
WHERE ROSHARE IN ('O', 'R');
```

RECOVER INDEX

Through DB2 V5, the RECOVER INDEX utility is used to re-create indexes from current data. RECOVER INDEX scans the table on which the index is based and regenerates the index

based on the actual data. Indexes are always recovered from actual table data, not from image copy and log data.

DB2 Version 6 changes the functionality of the RECOVER INDEX utility changes. Instead of rebuilding indexes from the current data, RECOVER INDEX will actually recover the index data by reading an image copy of the index data set. So, with DB2 V6, you can use the COPY utility to make backups of DB2 indexes and the RECOVER utility to restore them.

To provide equivalent functionality for re-creating an index from the current data, IBM provides a new utility called REBUILD INDEX. The REBUILD INDEX utility works exactly like RECOVER INDEX used to.

Organizations should begin changing all of their current RECOVER INDEX jobs to use REBUILD INDEX syntax instead. The REBUILD INDEX syntax is available in DB2 V5 and V4 (with PTF PQ09842) and will work exactly like RECOVER INDEX. After you migrate to DB2 V6, the RECOVER INDEX utility will cease to function if the proper index backup copies are not available to use during recovery.

Host Variables Without Colons

All DB2 programmers should know that host variables used in SQL statements in a program should be preceded by a colon. So, if a host variable is named HV it should be coded in the SQL statement as :HV. However, most programmers do not know that through V5, DB2 programs tolerate host variables that are not preceded by a colon. DB2 will spit out a warning message, but will process the SQL containing the offending host variable. This “feature” is no longer supported as of DB2 V6.

The reason IBM eliminated this feature is the rising complexity of SQL. It is getting too difficult for DB2 to differentiate host variables from SQL when it parses the SQL to be prepared for execution. With all of the new features being added to DB2, the rising complexity of the SQL language will continue unabated. As such, for DB2 V6 and onward, all host variables must be prefixed with a colon, or the statement will fail to execute.

This change should not impact many programs because most organizations have DB2 standards that dictate all host variables must begin with a colon. However, because DB2 has tolerated host variables without a colon for many years (through DB2 V5), you should inspect all DB2 SQL statements in application programs to ensure compliance prior to migrating to DB2 V6.

This is the most difficult problem to find and fix as a result of moving to DB2 Version 6. If you do not fix the problem prior to migrating to V6, any programs containing offending host variables will fail the next time they are rebound.

Dataset Passwords

The ability to provide security via dataset passwords was a little-used feature of DB2. Using the DSETPASS keyword of the CREATE TABLESPACE and CREATE INDEX statement, it was possible to password protect DB2 datasets.

This feature disappeared with DB2 V6. If you need to protect your DB2 datasets outside of DB2 security, you can use RACF, ACF2, Top Secret, or whatever security package you have installed at your site to accomplish this.

To find datasets that are password protected using DSETPASS, issue the following SQL statement:

```
SELECT 'INDEX ', CREATOR, NAME
FROM SYSIBM.SYSINDEXES
WHERE DSETPASS <> ' '
UNION ALL
SELECT 'TSPACE', DBNAME, NAME
FROM SYSIBM.SYSTABLESPACE
WHERE DSETPASS <> ' ';
```

Stored Procedure Registration

Prior to DB2 Version 6, after coding a stored procedure, you must register information about that stored procedure in the DB2 system catalog. This process is in sharp contrast to the manner in which other database objects are recorded in the system catalog. Typically, when an object is created, DB2 automatically stores the metadata description of that object in the appropriate DB2 catalog tables. For example, to create a new table, the CREATE TABLE statement is issued and DB2 automatically records the information in multiple system catalog tables (SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, SYSIBM.SYSTABLESPACE, and possibly SYSIBM.SYSFIELDS, SYSIBM.SYSCHECKS, SYSIBM.SYSCHECKDEP, SYSIBM.SYSRELS, and SYSIBM.SYSFOREIGNKEYS). Because stored procedures were not created within DB2, nor were they created using DDL, the database administrator had to use SQL INSERT statements to populate the SYSIBM.SYSPROCEDURES system catalog table with the metadata for the stored procedure.

The following SQL provides an example of an INSERT to register a stored procedure:

```
INSERT INTO SYSIBM.SYSPROCEDURES
  (PROCEDURE, AUTHID, LUNAME, LOADMOD, LINKAGE,
   COLLID, LANGUAGE, ASUTIME, STAYRESIDENT,
   IBMREQD, RUNOPTS, PARMLIST, RESULT_SETS,
   WLM_ENV, PGM_TYPE, EXTERNAL_SECURITY,
   COMMIT_ON_RETURN)
VALUES
  ('PROCNAME', ' ', ' ', 'LOADNAME', ' ',
   'COLL0001', 'COBOL', 0, 'Y',
   'N', ' ', 'NAME CHAR(20) INOUT', 1,
   ' ', 'M', 'N', 'N');
```

This SQL statement registers a stored procedure written in COBOL and named PROCNAME with a load module named LOADNAME. It uses a package with a collection ID of COLL0001. Any location can execute this procedure. The program stays resident and uses the DB2 SPAS (not Workload Manager), and no limit is set on the amount of time it can execute before being canceled. Furthermore, the stored procedure uses one input/output parameter, and the parameter cannot be null.

This method of registering stored procedures changed in DB2 V6. Instead of the `INSERT` statement, `CREATE` and `ALTER` statements are provided for registering stored procedures to the DB2 system catalog. Additionally, a new catalog table named `SYSIBM.SYSROUTINES` replaces `SYSIBM.SYSPROCEDURES`. This new table will store information on triggers, user-defined functions, and stored procedures. The metadata for all of these “routines” will be provided to the system catalog by means of DDL statements.

Many organizations have procedures for creating and updating stored procedures that include registration. These procedures will need to be modified to work with DB2 V6 and later releases.

Synopsis

Version 6 was the first release of DB2 to take features out of the product. As such, organizations had to understand what was being removed, know how to provide similar functionality with other DB2 features, and develop a plan to migrate away from the non-supported features. The sooner you remove the old technology, the sooner you can move to the latest and greatest version of DB2 that is available.

IN THIS CHAPTER

- DB2 Version 6 Features
- DB2 Version 5 Features
- DB2 Version 4 Features

WEB 2

Short Summary of DB2 V4 Through V6 Changes

This appendix provides short checklists of features for the most recent versions of DB2 prior to the two most-recent versions covered in this book. There have been four versions of DB2 released since 1995:

- DB2 Version 4 (also known as DB2 V4 or DB2 V4.1)
- DB2 Version 5 (also known as DB2 V5 or DB2 V5.1)
- DB2 Version 6 (also known as DB2 V6 or DB2 V6.1)
- DB2 Version 7 (also known as DB2 V7 or DB2 V7.1)
- DB2 Version 8 (also known as DB2 V8 or DB2 V8.1)

The following sections contain very short, bulleted lists that inventory the features of each release. The lists are in reverse chronological order.

DB2 Version 6 Features

DB2 V6 has been generally available since June 1999. The most important new features provided by V6 are listed in the following sections.

Database Administration Features

16-terabyte tables.

Object/relational capabilities including BLOBs, CLOBs, and DBCLOBs, triggers, UDFs, and UDTs.

Multimedia support with DB2 Extenders.

8K and 16K tablespace page sizes.

VARCHAR column resizing.

Explicit CREATE support for stored procedures.

Ability to specify a default buffer pool for indexes.

Enhanced support for pattern-matching characters in DB2 commands.

Improved partition rebalancing.

You can change checkpoint frequency dynamically using the SET LOG command.

Object code version of DSNTPE2 provided (no longer need a PL/I compiler).

Utility Features

COPY and RECOVER can process a list of objects in parallel and recover indexes and tablespaces at the same time from image copies and the log.

Parallel index build reduces the elapsed time of LOAD and REORG jobs involving more than one index.

Inline statistics collection during utility jobs.

Threshold limits to determine when to run REORG.

Remote site recovery improvements.

Programming Features

SQLJ support for embedded SQL in Java programs.

Three-part names support using DRDA.

Many stored procedure enhancements, including nested procedure CALLS and the ability to issue CALL statements dynamically using ODBC drivers.

More than 50 new built-in functions.

Up to 225 tables permitted in SQL SELECT, INSERT, UPDATE, and DELETE statements and views.

Support for VALUES and VALUES INTO.

Direct-row access using the ROWID data type to re-access a row directly without using the index or scanning the table.

ODBC extensions including new and modified APIs, support for DB2 V6 object/relational extensions, and ODBC catalog query redirection to shadow copies of DB2 catalog tables.

Performance Features

Optimization hints.

Predictive governing.

Statement cost estimation.

Buffer pools in data spaces.

DDF connection pooling.

Improved workload balancing in parallel Sysplex.

Faster log apply process.

Ability to postpone backout work during a restart.

Increased log output buffer size (from 1000 to 100000 4K buffers).

Query parallelism improvements.

DB2 Catalog Impact

9 new tables.

1 table no longer used, but kept for fallback purposes.

24 tables have one or more new or changed columns.

52 total new columns.

80 total revised columns.

DB2 Version 5 Features

DB2 V5 was first announced by IBM as V4.2. However, late in 1996 IBM changed plans and switched from a point release to a full-fledged new version. DB2 V5, generally available since June 1997, offers the following features.

Database Administration Features

LARGE tables (up to 254 partitions; approx. 1TB).

Multiple stored procedure address spaces.

Table renaming.

ASCII server support.

Native TCP/IP.

DCE security.

DDL-based support rows per page (up to 255).

Workstation GUI install.

Utility Features

Online REORG.

LOAD and REORG improvements.

COPY with thresholds.

RUNSTATS using sampling.

Programming Features

CASE expressions.

Stored procedure results sets.

Temporary tables.

RRSAF.

- Call Level Interface (ODBC).
- NULLIF function.
- STRIP function.
- Visual EXPLAIN.
- Temporary tables.

Performance Features

- Optimization changes.
- Skip partition scanning.
- Changes to stage 1 and indexable predicates.
- SQL caching.
- Persistent dynamic BIND.
- Query Sysplex parallelism.
- Partition locking.
- Data sharing improvements.

DB2 Catalog Impact

- Communications Database moved to the DB2 Catalog Database.
- 8 new or renamed tables.
- 31 tables have one or more new or changed columns.
- 65 total new columns.
- 59 total revised columns.

DB2 Version 4 Features

IBM introduced many new and useful features with DB2 V4. The highlights of this release are as follows.

Database Administration Features

- DB2 Catalog REORG.
- User-defined DB2 Catalog indexes.
- COPY, RECOVER, and REORG improvements.
- Dynamic SQL security improvements.
- REFERENCES privilege.
- Data sharing.
- Type 2 indexes.

- User-defined defaults.
- Check constraints.
- UNIQUE WHERE NOT NULL indexes.
- Row-level locking.
- Multi-character command prefixes.
- Tracking DFSMS concurrent copies in the DB2 Catalog.

Client/Server Features

- Stored procedures.
- Support for 25,000 distributed connections.

Performance Features

- Partition scanning (page range scan).
- Query CP parallelism.
- Uncommitted read (read-through locks).
- No locks on Type 2 indexes.

Programming Features

- Outer join.
- In-line views (nested tables).
- COALESCE function.
- Column renaming using AS.
- DCLGEN improvements.

DB2 Catalog Impact

- Communications Database moved to the DB2 Catalog Database.
- 3 new tables.
- 16 tables have one or more new or changed columns.
- 18 total new columns.
- 21 total revised columns.

- Basic Index Structure

Type 1 Indexes

The ability to create indexes on DB2 tables has been around since the first release of DB2. The first type of index that was available is now referred to as a type 1 index. Type 1 indexes were made obsolete when type 2 indexes were introduced in DB2 Version 4. However, IBM supported type 1 indexes through Version 5. As of DB2 V6, type 2 indexes can no longer be used by DB2.

Type 2 indexes are preferable to type 1 indexes because they eliminate index locking. Furthermore, most newer features of DB2 require type 2 indexes.

Basic Index Structure

Before examining the specifics of the layout of index data pages, let's first examine the basic structure of DB2 indexes.

A DB2 index is a modified *b-tree* (balanced tree) structure that orders data values for rapid retrieval. The values being indexed are stored in an inverted tree structure, as shown in Figure 1.

As values are inserted and deleted from the index, the tree structure is automatically balanced, realigning the hierarchy so that the path from top to bottom is uniform. This realignment minimizes the time required to access any given value by keeping the search paths as short as possible. To implement b-tree indexes, DB2 uses the following types of index data pages:

- | | |
|-----------------|---|
| Space map pages | Space map pages determine what space is available in the index for DB2 to utilize. |
| Root page | Only one root page is available per index. The root page must exist at the highest level of the hierarchy for every index structure. It can be structured as either a leaf or a non-leaf page, depending on the number of entries in the index. |

- Non-leaf pages Non-leaf pages are intermediate-level index pages in the b-tree hierarchy. Non-leaf pages need not exist. If they do exist, they contain pointers to other non-leaf pages or leaf pages. They never point to data rows.
- Leaf pages Leaf pages contain pointers to the data rows of a table. Leaf pages must always exist. In a single page index, the root page is a leaf page.

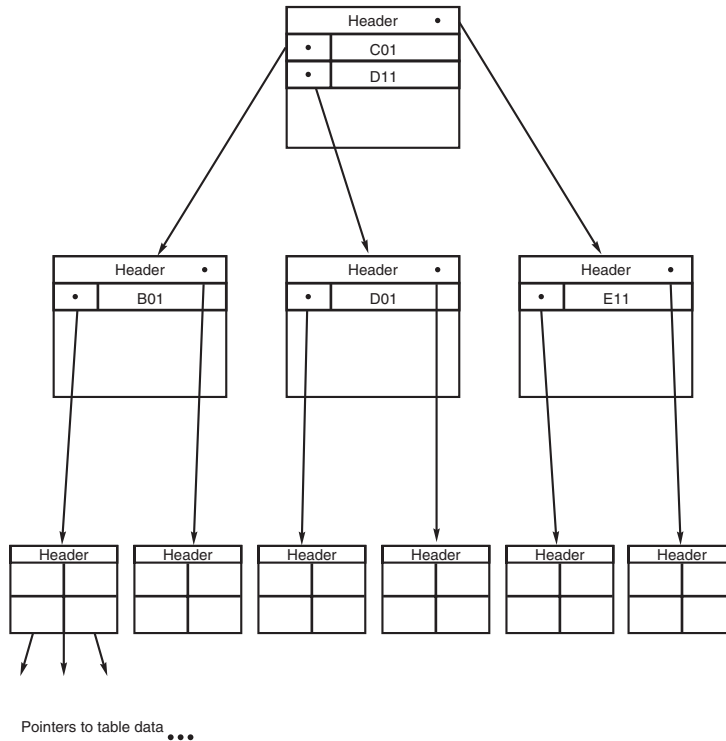


FIGURE 1 DB2 index structure.

The pointers in the leaf pages of an index are called a *record ID*, or *RID*. Each RID is a combination of the tablespace page number and the row pointer for the data value, which together indicate the location of the data value.

The level of a DB2 index indicates whether it contains non-leaf pages. The smallest DB2 index is a one-level index; the root page contains the pointers to the data rows. In this case, the root page is also a leaf page, and no non-leaf pages are available. This is true for Type 1 indexes only; no one-level Type 2 indexes exist. A two-level index does not contain non-leaf pages. The root page points directly to leaf pages, which in turn point to the rows containing the indexed data values.

A three-level index, such as the one shown in Figure 1, contains one level for the root page, another level for non-leaf pages, and a final level for leaf pages. The larger the number of levels for an index, the less efficient it will be. You can have any number of

intermediate non-leaf page levels. Try not to have indexes with more than three levels because they are generally very inefficient.

Type 1 Index Data Pages

Type 1 non-leaf pages are physically formatted as shown in Figure 2. Each non-leaf page contains the following:

- A 12-byte index page header that houses consistency and recoverability information for the index.
- A 16-byte physical header that stores control information for the index page. For example, the physical header controls administrative housekeeping such as the type of page (leaf or non-leaf), the location of the page in the index structure, and the ordering and size of the indexed values.
- A 17-byte logical header that stores additional consistency and recoverability checking information, as well as administers free space.

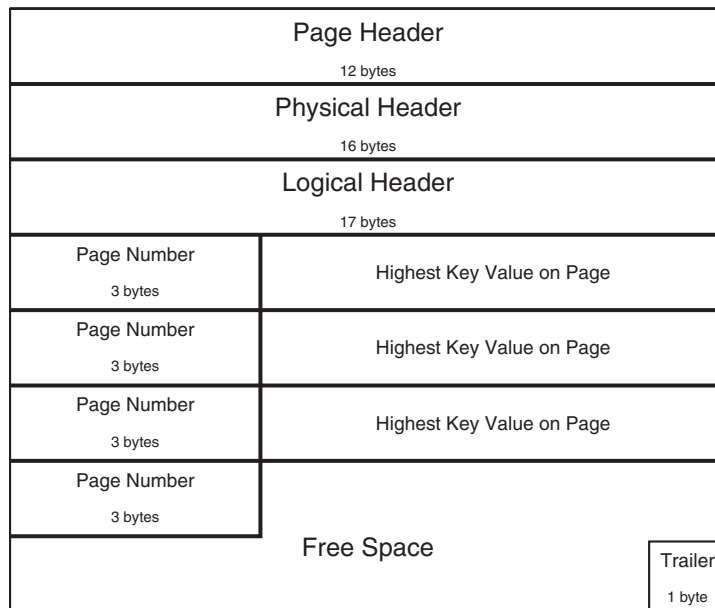


FIGURE 2 Type 1 index non-leaf page layout.

The physical structure of a type 1 index leaf page differs depending on the parameters specified when the index is created. Type 1 index pages can be broken down into smaller portions, known as *subpages*. A type 1 index can be defined as having 1, 2, 4, 8, or 16 subpages. The physical structure of type 1 index leaf pages depends on the number of subpages defined for the index.

For type 1 indexes, increasing the number of subpages can decrease contention, but this may decrease the efficiency of access to the index data. Specify `SUBPAGES 1` for infrequently updated type 1 indexed columns.

For a type 1 clustering index, you might want to try setting the number of subpages such that each subpage contains the same number of rows as the data pages of the tablespace. This can reduce locking of unrelated data. If the index is not clustered, do not attempt this, because the corresponding index subpages will contain different rows than the tablespace pages, and no gain in performance will be realized.

Refer to Figure 3 for the physical layout of a type 1 index leaf page with a subpage specification of 1. The page header, physical header, and logical header are used for the same purposes as they are in non-leaf pages. The remainder of the page is used for index entries. Each index entry is composed of indexed values and RID pointers to the table data.

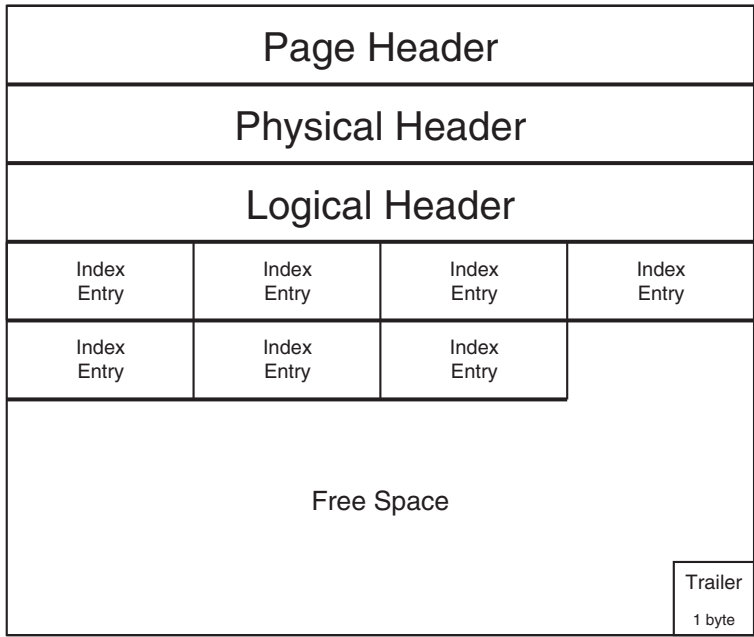


FIGURE 3 Layout of a type 1 index leaf page containing one subpage.

Refer to Figure 4 for the physical layout of a type 1 index leaf page with a subpage specification greater than 1. A subpage directory replaces the single logical header. This directory contains an array of pointers used to locate and administer the index subpages. Each subpage has its own logical header, allowing free space to exist on each subpage.

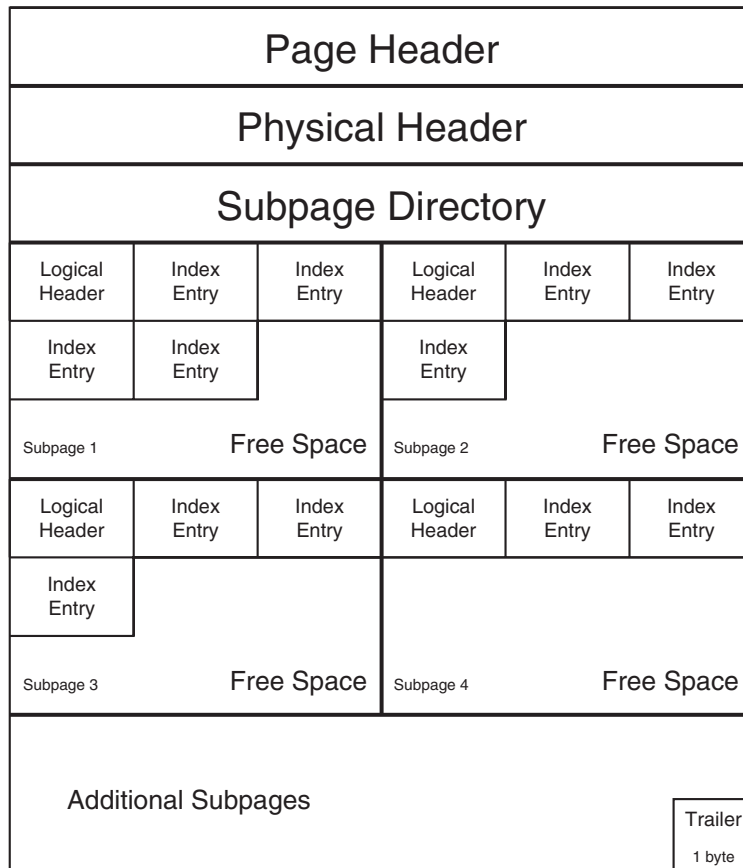


FIGURE 4 Layout of a type 1 index leaf page containing more than one subpage.

The final physical index structure to explore is the index entry. You can create both unique and non-unique indexes for each DB2 table. When the index key is of varying length, DB2 pads the columns to their maximum length, making the index keys a fixed length. A unique index contains entries, and each entry has a single RID. In a unique index, no two index entries can have the same value because the values being indexed are unique (see Figure 5).

Unique Index Entries

Index Key Value(s)	RID
--------------------	-----

Non-Unique Index Entries

Header	Index Key Value(s)	RID	RID	RID	RID
--------	--------------------	-----	-----	-----	-----

FIGURE 5 Index entries.

Synopsis

This appendix is provided for those shops that have not yet converted to DB2 V6 and still have type 1 indexes. No new indexes should be defined as type 1 and you should immediately begin to convert all type 1 indexes to type 2 indexes. This is important because type 1 indexes are no longer supported by DB2 as of Version 6.

APPENDIX A

DB2 Sample Tables

This appendix provides information on the DB2 sample tables used in most of the figures and examples in this book. These tables are used as examples in this book as a convenience because they are bundled with DB2, installed at most DB2 shops, and are generally available for everyone's use.

An understanding of the data in the sample tables and the relationship between these tables is imperative to understanding the SQL in this book. The DB2 sample tables primarily contain information about projects and the entities involved in working on these projects. Figure A.1 shows these entities and the relationships between them.

The sample tables represent departments, employees, projects, activities, activities assigned to a project, and employees assigned to a project's activities. In the following sections, you can find a general description of each table, its columns, and its relationship to the other sample tables, along with its table creation DDL.

The Activity Table: DSN8810.ACT

DSN8810.ACT describes activities that can be performed for projects. This table simply provides activity information. It does not tie each activity to a project. The following information about an activity is recorded: the activity number, the activity keyword, and the activity description. The activity number (ACTNO) is the primary key for this table.

DSN8810.ACT is a parent table for DSN8810.PROJACT. Two indexes have been built for this table: DSN8810.XACT1 is a primary key index on ACTNO, and DSN8810.XACT2 is a unique index on ACTKWD.

PART IX: APPENDIXES

APPENDIX A	DB2 Sample Tables
APPENDIX B	DB2 Tool Vendors
APPENDIX C	Valid DB2 Data Types
APPENDIX D	DB2 Limits
APPENDIX E	DB2 on Other Platforms
APPENDIX F	DB2 Version 7 Overview
APPENDIX G	DB2 Version 8 Overview
APPENDIX H	Reorganizing the DB2 Catalog

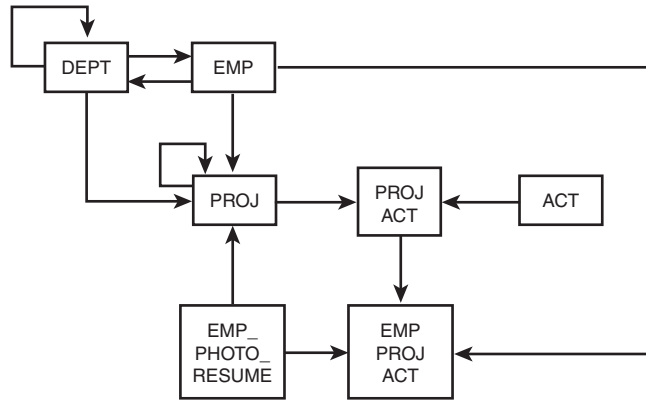


FIGURE A.1 DB2 sample table relationships.

DSN8810.ACT Table DDL

```

CREATE TABLE DSN8810.ACT
( ACTNO          SMALLINT          NOT NULL,
  ACTKWD         CHAR(6)           NOT NULL,
  ACTDESC       VARCHAR(20)       NOT NULL,
  PRIMARY KEY (ACTNO)
)
IN DSN8D81A.DSN8S81P
CCSID EBCDIC;

```

The Department Table: DSN8810.DEPT

DSN8810.DEPT describes information about departments that may be participating in projects. The following information is stored for each department: the department number, the department name, the employee number for the manager of this department, and the department number for the department to which this department reports. The department number is the primary key.

Referential integrity is used to implement a self-referencing constraint for ADMRDEPT. This referential constraint establishes the higher level department to which this department reports. A constraint also exists for MGRNO to EMPNO, the primary key of the DSN8810.EMP table. It ensures that the manager of a department is a valid employee.

Three indexes have been built for this table: DSN8810.XDEPT1 is a primary key index on DEPTNO, DSN8810.XDEPT2 is an index on MGRNO, and DSN8810.XDEPT3 is an index on ADMRDEPT.

DSN8810.DEPT Table DDL

```

CREATE TABLE DSN8810.DEPT
( DEPTNO        CHAR(3)           NOT NULL,
  DEPTNAME     VARCHAR(36)       NOT NULL,
  MGRNO        CHAR(6),
  ADMRDEPT     CHAR(3)           NOT NULL,

```

```

        LOCATION          CHAR(16),
        PRIMARY KEY (DEPTNO)
    )
    IN DSN8D81A.DSN8S81D
    CCSID EBCDIC;

ALTER TABLE DSN8810.DEPT
    FOREIGN KEY RDD (ADMRDEPT)
        REFERENCES DSN8810.DEPT ON DELETE CASCADE;

ALTER TABLE DSN8810.DEPT
    FOREIGN KEY RDE (MGRNO)
        REFERENCES DSN8810.EMP ON DELETE SET NULL;

```

The Employee Table: DSN8810.EMP

DSN8810.EMP describes employees in the organization. This table is in a partitioned table space. The following information is retained about employees: the employee's number, first name, middle initial, and last name; the department where this employee works; the employee's phone number; the date the employee was hired; and the employee's job description, education level, sex, birth date, salary, commission, and bonus data. The primary key is the employee number.

This table is a child of DSN8810.DEPT by the WORKDEPT column and a parent table for DSN8810.PROJ. Two indexes have been built for this table: DSN8810.XEMP1 is a primary unique, partitioned index on EMPNO, and DSN8810.XEMP2 is an index on WORKDEPT.

DSN8810.EMP Table DDL

```

CREATE TABLE DSN8810.EMP
    (EMPNO          CHAR(6)          NOT NULL,
     FIRSTNAME     VARCHAR(12)     NOT NULL,
     MIDINIT       CHAR(1)          NOT NULL,
     LASTNAME      VARCHAR(15)     NOT NULL,
     WORKDEPT      CHAR(3),
     PHONENO       CHAR(4) CONSTRAINT NUMBER CHECK
        (PHONENO >= '0000' AND
         PHONENO <= '9999'),
     HIREDATE      DATE,
     JOB           CHAR(8),
     EDLEVEL       SMALLINT,
     SEX           CHAR(1),
     BIRTHDATE     DATE,
     SALARY        DECIMAL(9,2),
     BONUS         DECIMAL(9,2),
     COMM          DECIMAL(9,2),
     PRIMARY KEY (EMPNO)
     FOREIGN KEY RED (WORKDEPT)
        REFERENCES DSN8810.DEPT ON DELETE SET NULL
    )
    EDITPROC DSN8EAE1
    IN DSN8D81A.DSN8S81E
    CCSID EBCDIC;

```

The Employee Photo & Resume Table: DSN8810.EMP_PHOTO_RESUME

DSN8810.EMP_PHOTO_RESUME contains photos and resume text for employees in the DSN8810.EMP table, previously described. The table contains a LOB column for the resume and two LOB columns for photos, one in PSEG format and one in BMP format.

The table is a parent table of DSN8810.PROJ with a foreign key on column RESPEMP. There are four indexes associated with the tables required to store photos and resumes: one on the base table and one each on the auxiliary tables. DSN8810.XEMP_PHOTO_RESUME is a primary unique index on the base table; and DSN8810.XAUX_BMP_PHOTO, DSN8810.XAUX_PSEG_PHOTO, DSN8810.XAUX_EMP_RESUME are each unique indexes on the respective auxiliary tables.

DSN8810.EMP_PHOTO_RESUME Table and Auxiliary Table DDL

```
CREATE TABLE DSN8810.EMP_PHOTO_RESUME
(EMPNO          CHAR(06) NOT NULL,
 EMP_ROWID      ROWID NOT NULL GENERATED ALWAYS,
 PSEG_PHOTO     BLOB(100K),
 BMP_PHOTO      BLOB(100K),
 RESUME         CLOB(5K))
PRIMARY KEY EMPNO
IN DSN8D81L.DSN8S81B
CCSID EBCDIC;
```

An auxiliary table is required for each LOB column in the table. The following DDL creates the auxiliary tables required for the three LOB columns in DSN8810.EMP_PHOTO_RESUME:

```
CREATE AUX TABLE DSN8810.AUX_BMP_PHOTO
IN DSN8D61L.DSN8S61M
STORES DSN8810.EMP_PHOTO_RESUME
COLUMN BMP_PHOTO;

CREATE AUX TABLE DSN8810.AUX_PSEG_PHOTO
IN DSN8D61L.DSN8S61L
STORES DSN8810.EMP_PHOTO_RESUME
COLUMN PSEG_PHOTO;

CREATE AUX TABLE DSN8810.AUX_EMP_RESUME
IN DSN8D61L.DSN8S61N
STORES DSN8810.EMP_PHOTO_RESUME
COLUMN RESUME;
```

Each auxiliary table must have a unique index defined on it. DSN8810.AUX_BMP_PHOTO has a required unique index named DSN8810.XAUX_BMP_PHOTO; DSN8810.AUX_PSEG_PHOTO has a required unique index named DSN8810.XAUX_PSEG_PHOTO; and DSN8810.AUX_EMP_RESUME has a required unique index named DSN8810.XAUX_EMP_RESUME.

The Employee Assignment Table: DSN8810.EMPPROJECT

DSN8810.EMPPROJECT details which employee performs which activity for each project. It effectively records the assignment of employees to a given activity for a given project. To accomplish this assignment, the table stores an employee number, a project number, and

an activity number on every row, along with information about this employee's assignment. This additional information consists of the percentage of time the employee should spend on this activity, the date the activity starts, and the date the activity ends. No primary key is implemented, but a unique index is used on the combination of PROJNO, ACTNO, EMSTDATE, and EMPNO.

The table is a child of both DSN8810.PROJACT and DSN8810.EMP. Two indexes exist for this table: DSN8810.XEMPPROJACT1 is a unique index on PROJNO, ACTNO, EMSTDATE, and EMPNO; and DSN8810.XEMPPROJACT2 is an index on EMPNO.

DSN8810.EMPPROJACT Table DDL

```
CREATE TABLE DSN8810.EMPPROJACT
(EMPNO          CHAR(6)          NOT NULL,
 PROJNO         CHAR(6)          NOT NULL,
 ACTNO          SMALLINT        NOT NULL,
 EMPTIME        DECIMAL(5,2),
 EMSTDATE       DATE,
 EMENDATE       DATE,
 FOREIGN KEY REPAPA (PROJNO, ACTNO, EMSTDATE)
   REFERENCES DSN8810.PROJACT ON DELETE RESTRICT,
 FOREIGN KEY REPAE (EMPNO)
   REFERENCES DSN8810.EMP ON DELETE RESTRICT
)
IN DSN8D81A.DSN8S81P
CCSID EBCDIC;
```

The Project Table: DSN8810.PROJ

DSN8810.PROJ defines all the projects for the organization. It contains information on the project's number; the project's name; the responsible department number and employee number; the project's staffing requirements, start date, and end date; and the project number of any related, superior project. The primary key is PROJNO.

DSN8810.PROJ is a self-referencing table because one project can relate to another by the MAJPROJ column, which identifies a parent project. It is also a parent table because it has relationships to DSN8810.DEPT for the responsible department and to DSN8810.EMP for the responsible employee.

Two indexes exist for this table: DSN8810.XPROJ1 is a primary key index on PROJNO, and DSN8810.XPROJ2 is an index on RESPEMP.

DSN8810.PROJ Table DDL

```
CREATE TABLE DSN8810.PROJ
(PROJNO          CHAR(6) PRIMARY KEY NOT NULL,
 PROJNAME        VARCHAR(24)          NOT NULL WITH DEFAULT
   'PROJECT NAME UNDEFINED',
 DEPTNO          CHAR(3)              NOT NULL
   REFERENCES DSN8810.DEPT ON DELETE RESTRICT,
 RESPEMP         CHAR(6)              NOT NULL
   REFERENCES DSN8810.EMP ON DELETE RESTRICT,
 PRSTAFF         DECIMAL(5, 2),
 PRSTDATE        DATE,
```

```

    PRENDATE      DATE,
    MAJPROJ       CHAR(6)
)
IN DSN8D81A.DSN8S81P
CCSID EBCDIC;

ALTER TABLE DSN8810.PROJ
    FOREIGN KEY RPP (MAJPROJ)
    REFERENCES DSN8810.PROJ ON DELETE CASCADE;
```

The Project Activity Table: DSN8810.PROJACT

DSN8810.PROJACT records the activities for each project. It stores the following information: the project's number, the activity's number, the number of employees needed to staff the activity, and the estimated activity start date and end date.

DSN8810.PROJACT is a parent of the DSN8810.EMPPROJACT table and functions as a child table for DSN8810.ACT and DSN8810.PROJ. This table has one index: DSN8810.XPROJAC1 is a unique primary key index on PROJNO, ACTNO, and ACSTDATE.

DSN8810.PROJACT Table DDL

```

CREATE TABLE DSN8810.PROJACT
    (PROJNO      CHAR(6)          NOT NULL,
     ACTNO       SMALLINT        NOT NULL,
     ACSTAFF     DECIMAL(5,2),
     ACSTDATE    DATE            NOT NULL,
     ACENDATE    DATE,
     MAJPROJ     CHAR(6),
     PRIMARY KEY (PROJNO, ACTNO, ACSTDATE),
     FOREIGN KEY RPAP (PROJNO)
       REFERENCES DSN8810.PROJ ON DELETE RESTRICT,
     FOREIGN KEY RPAA (ACTNO)
       REFERENCES DSN8810.ACT ON DELETE RESTRICT
)
IN DSN8D81A.DSN8S81P
CCSID EBCDIC;
```

The Sample STOGROUP

The storage group used by the sample database is DSN8G410. The following statement is provided by IBM to define the sample STOGROUP. (Of course, the VOLUMES, VCAT, and PASSWORD information are usually modified prior to the creation of the storage group.)

```

CREATE STOGROUP DSN8G810
    VOLUMES (DSNV01)
    VCAT     DSN8C810;;
```

Sample Databases and Table Spaces

Tables A.1 and A.2 provide a synopsis of the databases and table spaces used for the sample tables.

TABLE A.1 Sample Databases

Database Name	Storage Group	Buffer Pool	CCSID
DSN8D81A	DSN8G810	BP0	EBCDIC
DSN8D81L	DSN8G810	BP0	EBCDIC
DSN8D81P	DSN8G810	BP0	EBCDIC

TABLE A.2 Sample Table Spaces

Table Space Name	Database Name	Buffer Pool	Table Space Type	Lock Size	Compressed
DSN8S81B	DSN8D81L	BP0	SIMPLE	PAGE	NO
DSN8S81C	DSN8D81P	BP0	SEGMENTED	TABLE	NO
DSN8S81D	DSN8D81A	BP0	SIMPLE	PAGE	NO
DSN8S81E	DSN8D81A	BP0	PARTITIONED	PAGE	YES
DSN8S81P	DSN8D81A	BP0	SEGMENTED	ROW	NO
DSN8S81R	DSN8D81A	BP0	SIMPLE	PAGE	NO
DSN8S81S	DSN8D81A	BP0	SIMPLE	PAGE	NO
DSN8S81L	DSN8D81L	BP0	LOB	N/A	N/A
DSN8S81M	DSN8D81L	BP0	LOB	N/A	N/A
DSN8S81N	DSN8D81L	BP0	LOB	N/A	N/A

Views on the Sample Tables

When you install the sample database, there are also several views created on the sample tables. These views are listed in Table A.3. All of the views are qualified by DSN8810 (where the fifth letter indicates the version of DB2, in this case, 8 for V8).

TABLE A.3 Sample Views

View	Table(s)	Columns	Predicates?
VDEPT	DEPT	DEPTNO, DEPTNAME, MGRNO, ADMRDEPT	N/A
VHDEPT	DEPT	DEPTNO, DEPTNAME, MGRNO, ADMRDEPT, LOCATION	N/A
VEMP	EMP	FIRSTNME, MIDINIT, LASTNAME, WORKDEPT	N/A
VEMPLP	EMP	EMPNO, PHONENO	N/A
VPROJ	PROJ	PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTAFF, PRSTDATE, PRENDATE, MAJPROJ	N/A
VACT	ACT	ACTNO, ACTKWD, ACTDESC	N/A
VPROJACT	PROJACT	PROJNO, ACTNO, ACSTAFF, ACSTDATE, ACENDATE	N/A
VEMPPROJACT	EMPPROJACT	EMPNO, PROJNO, ACTNO, EMPTIME, EMSTDATE, EMENDATE	N/A

TABLE A.3 Continued

View	Table(s)	Columns	Predicates?
VDEPMG1	DEPT	DEPTNO, DEPTNAME, MGRNO ADMRDEPT	MGRNO = EMPNO
VEMPDPT1	EMP DEPT	FIRSTNME, MIDINIT, LASTNAME DEPTNO, DEPTNAME, MGRNO	MGRNO = EMPNO
	EMP	SUBSTR(FIRSTNME, 1, 1), MIDINIT, LASTNAME, WORKDEPT	
VPHONE	EMP	LASTNAME, FIRSTNME, MIDINIT, PHONENO, EMPNO,	WORKDEPT = DEPTNO
VPROJRE1	DEPT PROJ	DEPTNAME PROJNO, PROJNAME, DEPTNO MAJPROJ	RESPEMP = EMPNO
	EMP	EMPNO, FIRSTNME, MIDINIT, LASTNAME	
VSTAFAC1	PROJACT	PROJNO, ACTNO, ACSTAFF, ACSTDATE, ACENDATE,	ACTNO = ACTNO
	ACT	ACTDESC	
VSTAFAC2	EMPPROJACT	PROJNO, ACTNO, ACSTAFF, ACSTDATE, ACENDATE, EMPTIME, EMSTDATE, EMENDATE	ACTNO = ACTNO
	ACT	ACTDESC	
	EMP	EMPNO, FIRSTNME, MIDINIT, LASTNAME	EMPNO = EMPNO

There are also several views created on top of the sample views. The DDL for these views follows:

```
CREATE VIEW DSN8810.VASTRDE1
```

```
(DEPT1NO, DEPT1NAM, EMP1NO, EMP1FN, EMP1MI, EMP1LN, TYPE2,  
DEPT2NO, DEPT2NAM, EMP2NO, EMP2FN, EMP2MI, EMP2LN)
```

```
AS SELECT ALL
```

```
D1.DEPTNO, D1.DEPTNAME, D1.MGRNO, D1.FIRSTNME, D1.MIDINIT,  
D1.LASTNAME, '1',  
D2.DEPTNO, D2.DEPTNAME, D2.MGRNO, D2.FIRSTNME, D2.MIDINIT,  
D2.LASTNAME
```

```
FROM DSN8810.VDEPMG1 D1,  
DSN8810.VDEPMG1 D2
```

```
WHERE D1.DEPTNO = D2.ADMRDEPT;
```

```
CREATE VIEW DSN8810.VASTRDE2
```

```
(DEPT1NO, DEPT1NAM, EMP1NO, EMP1FN, EMP1MI, EMP1LN, TYPE2,  
DEPT2NO, DEPT2NAM, EMP2NO, EMP2FN, EMP2MI, EMP2LN)
```

```
AS SELECT ALL
```

```
D1.DEPTNO, D1.DEPTNAME, D1.MGRNO, D1.FIRSTNME, D1.MIDINIT,  
D1.LASTNAME, '2',  
D1.DEPTNO, D1.DEPTNAME, E2.EMPNO, E2.FIRSTNME, E2.MIDINIT,  
E2.LASTNAME
```

```

FROM DSN8810.VDEPMG1 D1,
     DSN8810.EMP E2
WHERE D1.DEPTNO = E2.WORKDEPT;

CREATE VIEW DSN8810.VPSTRDE1
  (PROJ1NO, PROJ1NAME, RESP1NO, RESP1FN, RESP1MI, RESP1LN,
   PROJ2NO, PROJ2NAME, RESP2NO, RESP2FN, RESP2MI, RESP2LN)
AS SELECT ALL
   P1.PROJNO, P1.PROJNAME, P1.REPEMP, P1.FIRSTNME, P1.MIDINIT,
   P1.LASTNAME,
   P2.PROJNO, P2.PROJNAME, P2.REPEMP, P2.FIRSTNME, P2.MIDINIT,
   P2.LASTNAME
FROM DSN8810.VPROJRE1 P1,
     DSN8810.VPROJRE1 P2
WHERE P1.PROJNO = P2.MAJPROJ;

CREATE VIEW DSN8810.VPSTRDE2
  (PROJ1NO, PROJ1NAME, RESP1NO, RESP1FN, RESP1MI, RESP1LN,
   PROJ2NO, PROJ2NAME, RESP2NO, RESP2FN, RESP2MI, RESP2LN)
AS SELECT ALL
   P1.PROJNO, P1.PROJNAME, P1.REPEMP, P1.FIRSTNME, P1.MIDINIT,
   P1.LASTNAME,
   P1.PROJNO, P1.PROJNAME, P1.REPEMP, P1.FIRSTNME, P1.MIDINIT,
   P1.LASTNAME
FROM DSN8810.VPROJRE1 P1
WHERE NOT EXISTS
  (SELECT *
   FROM DSN8810.VPROJRE1 P2
   WHERE P1.PROJNO = P2.MAJPROJ);

CREATE VIEW DSN8810.VFORPLA
  (PROJNO, PROJNAME, REPEMP, PROJDEP, FRSTINIT, MIDINIT, LASTNAME)
AS SELECT ALL
   F1.PROJNO, PROJNAME, REPEMP, PROJDEP,
   SUBSTR(FIRSTNME,1,1), MIDINIT, LASTNAME
FROM DSN8810.VPROJRE1 F1 LEFT OUTER JOIN DSN8810.EMPPROJACT F2
ON   F1.PROJNO = F2.PROJNO;

```

All the views outlined in Table A.3 and the previous DDL are used in the sample applications that are shipped with DB2.

IN THIS APPENDIX

- The Vendor List

APPENDIX B

DB2 Tools Vendors

This appendix contains an extensive listing of vendors who provide DB2 products. This list is not intended to be exhaustive, but lists the major players in the DB2 add-on tool market. It is accurate as of the writing of this book, but the software industry is dynamic; software development companies are buying out one another or selling their assets almost weekly.

Product names are not provided as names frequently change and some tools provide more than one function. Often these vendors supply software tools to manage other system software (such as CICS or IMS), but the focus of this list is on the DB2 development and management tools only. This list should be used as a reference tool; it does not offer any recommendations. Evaluation of the vendor's offerings is left to the reader and his or her organization.

The type of DB2 add-on tools each company supplies accompanies the vendor contact information. The tool types are coded based on the abbreviations used in Chapter 39, "Components of a Total DB2 Solution." The abbreviations are repeated here for reference:

ALT	Table alter tools
AUD	Auditing tools
CAT	DB2 Catalog query and analysis tools
COM	Compression tools
C/S	DB2-related client/server tools
DBA	Database analysis tools
DES	Database modeling and design tools
DSD	DASD and space management tools
EDT	DB2 table editors
ETL	Extract, Transform, Load tools (data movement)
INT	Data and referential integrity tools
MIG	DB2 object migration tools

MSC	Miscellaneous tools
NET	Internet, intranet, and Web enabling tools
OPR	Operational support tools
PC	PC-based DB2-related products
PLN	Plan analysis tools
PM	Performance monitors
PRF	Products to optimize and enhance performance
PRG	DB2 programming and development tools
QMF	QMF enhancement tools
QRY	Query tools
REP	Repositories and data dictionaries
SEC	Security tools
UTL	Utility enhancement, generation, and management tools

Keep in mind that some tools provide features that support more than one tool category. At a high level, it is wise to organize your evaluations of DB2 tools by tool category. Then concentrate on only the features of each tool integral to the category you are evaluating. This is the recommended approach to DB2 tool evaluation because many tools support multiple features. For example, an alter tool could also provide table editing capability. If you are evaluating alter capabilities and do not need table editing, do not let the additional feature of table editing influence your decision. Judge products based solely on the features you need. It is usually less costly (in the long run) to purchase two tools that fully support the required features (for example, altering and editing) than to purchase a single tool that only partially supports two (or more) capabilities.

This does not mean that tools that integrate multiple features always provide fewer capabilities than single-function tools. One integrated tool could provide all the features a small shop needs. Just be sure that a product supports your basic needs before looking at its additional “bells and whistles.” Additionally, it can make sense to purchase multiple tools from a single vendor because it will be more likely that the tools are similarly designed and offer some level of integrated functionality.

In general, it is wise to realize that third-party add-on tools can significantly improve the efficiency of DB2 application development and database administration. When evaluating products, look for features important to your organization. Consider adopting checklists for product comparisons based upon the features discussed in this article. And remember, although DB2 is a fantastic DBMS, it leaves quite a bit to be desired in the administration, data access, performance monitoring, and application development areas.

The following list is by no means comprehensive, but it does provide most of the medium- to large-sized companies that offer DB2 add-on tools and solutions.

The Vendor List

Vendor/Address

Ambeo Systems
5373 North Union Blvd.
Suite 201
Colorado Springs, CO 80918
(719) 548-7400
fax: (719) 548-8982
<http://www.ambeo.com>

Aonix
595 Market St.
10th Floor
San Francisco, CA 94105
(415) 543-0900
fax: (415) 543-0145
<http://www.aonix.com>

Ascential Software
50 Washington Street
Westboro, MA 01581
(508) 366-3888
<http://www.ascential.com>

ASG Software Solutions
1333 Third Avenue South
Naples, FL 34102
(239) 435-2200
fax: (239) 263-3692
<http://www.asg.com>

BEZ Systems, Inc.
222 W Adams St
Suite 1010
Chicago, IL 60606
(312) 641-2200
fax: (312) 641-1228
<http://www.bez.com>

BMC Software
2101 CityWest Blvd.
Houston, TX 77042
(713) 918-8800
fax: (713) 918-8000
<http://www.bmc.com>

Product Categories

PRF, ETL

C/S, DES, PRG, QRY

C/S, ETL, PRG

PM, PRG, REP, TST

PRF

ALT, AUD, CAT, COM, C/S, DBA, DSD, EDT,
INT, MIG, MSC, OPR, PLN, PM, PRF, SEC,
UTL

Brio Technology 3950 Fabian Way Suite 200 Palo Alto, CA 94303 (415) 856-8000 fax: (415) 856-8020 http://www.brio.com	C/S, QRY
Bunker Hill 501 Delancey Street Suite 609 San Francisco, CA 94107 (415) 512-0689 fax: (925) 314-9196 http://www.bunkerhill.com/	C/S, INT, PRG
Business Objects, Inc. 2870 Zanker Rd. San Jose, CA 95134 (408) 953-6000 fax: (408) 953-6001 http://www.businessobjects.com	C/S, QRY
Candle Corporation 2425 Olympic Blvd. Santa Monica, CA 90404 (310) 829-5800 fax: (310) 582-4287 http://www.candle.com IBM announced its intention to acquire Candle Corporation in April 2004	CAT, DBA, DSD, MIG, PLN, PM, PRG
CDB Software Inc. P.O. Box 771624 Houston, TX 77215 (713) 780-2382 fax: (713) 784-1842 http://www.cdbsoftware.com	DBA, PRG, UTL
CHARONWARE, s.r.o. Ulehlova 267/5 Ostrava, 700 30 Czech Republic http://www.casestudio.com/enu/default.aspx	DES
Chicago-Soft Products Ltd. 1 Maple Street Hanover, NH 03755 (603) 643-4002 Fax: (603) 643-4571 http://www.chicago-soft.com	OPR

Cognos Inc.
3775 Riverside Drive
P.O. Box 9707, Station T
Ottawa, ON
Canada K1G 4K9
(613) 738-1440
fax: (613) 738-0002
<http://www.cognos.com>

C/S, MSC, PRG, QRY

Computer Associates
One Computer Associates Plaza
Islandia, NY 11749
(631) 342-6000
fax: (631) 342-6800
<http://www.cai.com>

ALT, AUD, CAT, COM, C/S, DBA, DES, DSD,
EDT, INT, MIG, ETL, MSC, OPR, PLN, PM,
PRF, PRG, QRY, REP, SEC, UTL

Compuware Corporation
31440 Northwestern Highway
Framington Hills, MI 48334
(248) 737-7300
fax: (248) 737-7119
<http://www.compuware.com>

ALT, CAT, C/S, DBA, EDT, INT, MIG, MSC,
OPR, PM, PRG, SEC

CoSORT (IRI, Inc.)
2194 Highway A1A
Suite 303, Atlantis Center
Melbourne, FL 32937-4932
(321) 777-8889
<http://www.cosort.com>

ETL

Cross Access Corp.
2900 Gordon Avenue, Suite 100
Santa Clara, CA 95051
(408) 735-7545
fax: (630) 954-0554
<http://www.crossaccess.com>

C/S, ETL

Crystal Decisions
895 Emerson Street
Palo Alto, CA 94301-2413
(604) 681-3435
fax: (604) 681-2934
<http://www.crystaldecisions.com/>

QRY

Data Junction
2201 Northland Drive
Austin, TX 78756
(512) 452.6105
fax: (512) 459-1309
<http://www.datajunction.com>

ETL

NOTE

Data Junction was acquired by Pervasive Software in 2003. Pervasive's Web site is <http://www.pervasive.com>.

<p>DBE Software, Inc. 7601 Lewisville Rd., Suite 200 McLean, VA 22102 (703) 847-9500 fax: (703) 556-0089 http://www.dbesoftware.com</p>	<p>DES, MSC</p>
<p>Direct Computer Resources (631) 912-9025 fax: (631) 912-9029 http://www.datavantagedcr.com/</p>	<p>PRG</p>
<p>Embarcadero Technologies 425 Market Street, Suite 425 San Francisco, CA 94105 (415) 834-3131 fax: (415) 434-1721 http://www.embarcadero.com</p>	<p>CAT, DBA, DES, PRF, PRG</p>
<p>Foedero Technologies Inc. 1099 Kingston Road Suite 202, Pickering Ontario L1V 1B5 CANADA (905) 839-9815 fax:(905) 839-9251 http://www.foedero.com</p>	<p>PRF, PRG</p>
<p>GoldenGate Software, Inc. 3 Harbor Drive, Suite 200 Sausalito, CA 94965 (415) 289-8600 fax: (415) 289-8630 http://www.goldendate.com</p>	<p>ETL</p>
<p>Gupta Technologies, LLC 975 Island Drive Redwood Shores, CA 94065 USA (650) 596-3400 fax: (650) 596-4690 http://www.guptaworldwide.com</p>	<p>C/S, PC, PRG, QRY</p>

HiT Software
4020 Moorpark Avenue
Suite 100
San Jose, CA 95117
(408) 345-4001
fax: (408) 345-4899
<http://www.hitsw.com>

C/S, NET, PRG

HLS Technologies, Inc.
3322 Sturbridge Lane
Sugar Land, TX 77479
(281) 494-0971
fax: (281) 265-3006
<http://www.hlstechnologies.com>

INT, PRF

IBM Corporation
Santa Teresa Laboratory
555 Bailey Ave.
San Jose, CA 95141
(800) 426-4785
fax: (800) 426-4522
<http://www.software.ibm.com/data>

CAT, C/S, DBA, DES, EDT, ETL, MSC, NET,
PM, PRG, QMF, QRY, REP, UTL

IMSI
4720 Little John Trail
Sarasota, FL 34232
(800) 354-4674
fax: (941) 377-8475
<http://www.imsi-intl.com>

PLN, PRF

Informatica
2100 Seaport Boulevard
Redwood City, CA 94063
(650) 385-5000
fax: (650) 385-5500
<http://www.informatica.com>

ETL

Information Builders Inc.
Two Penn Plaza
New York, NY 10121
(212) 736-4433
fax: (212) 967-6406
<http://www.ibi.com>

ETL, PRG, QRY

Infospace
601 108th Avenue NE, Suite 1200
Bellevue, WA 98004
(425) 201-6100
fax: (425) 201-6150
<http://www.infospace-inc.com>

NET

Infotel Corporation P.O. Box 5158 Gulfport, FL 33737 (727) 343-5958 fax: (813) 674-0686 http://www.infotelcorp.com	COM, DSD, ETL, OPR, PRG, UTL
Magna Solutions Glockengiesserwall 26 20095 Hamburg Germany (49) 40-3010-4101 fax: (49) 40-3010-4257 http://www.silverrun.com/	DES
ManTech Systems Solutions 2700 South Quincy St. 4th Floor Arlington, VA 22206 (352) 337-0981 fax: (703) 671-9515 http://www.mssc-mantech.com/index1.html	ETL
Micro Focus 2465 E. Bayshore Rd. Palo Alto, CA 94303 (415) 856-4161 fax: (415) 856-6134 http://www.microfocus.com	C/S, PC, PRG
MicroStrategy, Inc. 1861 International Drive McLean, VA 22102 (703) 848-8600 fax: (703) 848-8610 http://www.microstrategy.com/	C/S, QRY
NEON Systems, Inc. 14141 SouthWest Freeway Suite 6200 Houston, TX 77478 (281) 491-4200 fax: (281) 242-3880 http://www.neonsys.com	C/S, ETL, NET, UTL

NetManage, Inc. 10725 North De Anza Blvd. Cupertino, CA 95014-2030 (408) 973-7171 fax: (408) 257-6405 http://www.netmanage.com	C/S, NET, PRG, QRY
OuterBay Technologies 20400 Stevens Creek Blvd. Suite 500 Cupertino, CA 95014-2296 (408) 340-1200 http://www.bitbybit.co.uk/	DES, INT
Praxis International 245 Winter St. Waltham, MA 02154-8716 (617) 622-5757 fax: (617) 622-5766 http://www.praxisint.com	ETL
Princeton SOFTECH 1060 State Rd. Princeton, NJ 08540-1423 (609) 497-0205 fax: (609) 497-0302 http://www.princetonsofttech.com	EDT, ETL, INT
Prolifics 116 John Street New York, NY 10038 (212) 267-7722 fax: (212) 608-6753 http://www.prolifics.com	C/S, INT, PRG
Quest Software 8001 Irvine Center Drive Irvine, CA 92618 (949) 754-8000 Fax: (949) 754-8999 http://www.quest.com	DBA, OPR, PRF, PM
Sagent Technology, Inc. 800 W. El Camino Real, 3rd Floor Mountain View, CA 94040 (650) 815-3100 Fax: (650) 815-3500 http://www.sagent.com/	ETL

Relational Architects Inc. 33 Newark St. Hoboken, NJ 07030 (201) 420-0400 fax: (201) 420-4080 http://www.relarc.com	OPR, PM, PRG, QMF
Responsive Systems Co. 281 Highway 79 Morganville, NJ 07751 (908) 972-1261 fax: (908) 972-9416 http://www.responsivesystems.com	DSD, OPR, PM
SAS Institute Inc. SAS Campus Drive Cary, NC 27513 (919) 677-8200 fax: (919) 677-8123 http://www.sas.com	PRG, QRY
SEGUS Inc. 12007 Sunrise Valley Drive Reston, VA 20191-3446 (703) 391-9650 fax: (703) 391-7133 http://www.segus.com/	CAT, EDT, RI
Softbase Systems Inc. 1664 Hendersonville Highway Asheville, NC 28803 (704) 277-9900 fax: (704) 277-9900 http://www.softbase.com	OPR, PRG
Starware Software 2150 Shattuck Ave., Suite 600 Berkeley, CA 94704 (510) 704-2000 fax: (510) 704-2001 http://www.starware.com	C/S
Striva Corporation 100 Enterprise Way Scotts Valley, CA 95066 (831) 438-8300 fax: 831-438-1990 http://www.striva.com	ETL

Sybase Corporation
6475 Christie Ave.
Emeryville, CA 94608
(510) 922-3555
fax: (510) 658-9441
<http://www.sybase.com>

DES, ETL, PRG, QRY

Tone Software Corp.
1735 S. Brookhurst Ave.
Anaheim, CA 92804
(714) 991-9460
fax: (714) 991-1831
<http://www.tonesoft.com>

COM, OPR

Treehouse Software
400 Broad St.
Sewickley, PA 15143
(412) 741-1677
fax: (412) 741-7245
<http://www.treehouse.com>

C/S, ETL

APPENDIX **C**

Valid DB2 Data Types

Data Type	Physical Storage	Value Range	COBOL Picture
SMALLINT	2 bytes	-32,768 to +32,767	PIC S9(4) COMP
INTEGER	4 bytes	-2,147,483,648 to +2,147,483,647	PIC S9(9) COMP
REAL	4 bytes	5.4E -79 to 7.2E+75	PIC USAGE COMP -1
FLOAT(1..21)	4 bytes	5.4E -79 to 7.2E+75	PIC USAGE COMP -1
DOUBLE PRECISION	8 bytes	5.4E -79 to 7.2E+75	PIC USAGE COMP -2
FLOAT(22..53)	8 bytes	5.4E -79 to 7.2E+75	PIC USAGE COMP -2
DECIMAL(<i>m,n</i>)	(<i>m/2</i>)+1 bytes	1 -10 ³¹ to 10 ³¹ -1	PIC S9(<i>m-n</i>)V9(<i>n</i>) COMP -3
CHARACTER(<i>n</i>)	<i>n</i> bytes	254 chars maximum	PIC X(<i>n</i>)
VARCHAR(<i>n</i>)	2 to <i>n</i> +2 bytes	4,046 bytes maximum 32,704 for 32KB pages	Ø1 VARCHAR. 49 LTH PIC S9(4)COMP. 49 COLUMN PIC X(<i>n</i>).
GRAPHIC(<i>n</i>)	2 <i>n</i> bytes	127 double-byte characters maximum	PIC G(<i>n</i>) DISPLAY -1
VARGRAPHIC(<i>n</i>)	2 to 2 <i>n</i> +2 bytes	2,023 double-byte characters maximum 32,704 for 32KB pages	Ø1 VGRAPHIC. 49 LENGTH PIC S9(4) 49 COLUMN PIC G(<i>n</i>) DISPLAY -1
DATE	4 bytes	0001-01-01 to 9999-12-31	PIC X(10)
TIME	3 bytes	00.00.00 to 24.00.00	PIC X(8)
TIMESTAMP	10 bytes	0001-01-01.00.00.00.000000 to 9999-12-31.24.00.00.000000	PIC X(26)
ROWID	up to 40 bytes	internal identifier	Ø1 ROWID -VAR USAGE IS SQL USAGE IS ROWID
BLOB	varies	up to 2GB	Ø1 BLOB -VAR USAGE IS SQL TYPE IS BLOB(<i>n</i>).

Data Type	Physical Storage	Value Range	COBOL Picture
- <i>or</i> -			01 BLOB-LOC USAGE IS SQL TYPE IS BLOB-LOCATOR.
CLOB	<i>varies</i>	up to 2GB	01 CLOB-VAR USAGE IS SQL TYPE IS CLOB(<i>n</i>).
- <i>or</i> -			01 CLOB-LOC USAGE IS SQL TYPE IS CLOB-LOCATOR.
DBCLOB	<i>varies</i>	up to 2GB	01 DBCLOB-VAR USAGE IS SQL TYPE IS DBCLOB(<i>n</i>).
- <i>or</i> -			01 DBCLOB-LOC USAGE IS SQL TYPE IS DBCLOB-LOCATOR.

NOTE

Applications that access or manipulate LOB data require either declared host variables to hold the LOB data or LOB locator variables to point to the LOB data.

DB2 will generate a PIC S9(9) USAGE IS BINARY field to be used for LOB locators defined as shown earlier.

For BLOB, CLOB, and DBCLOB, host variables defined for DB2 will generate a field structure to hold the LOB data. The first component is a PIC 9(9) COMP field to hold the length of the LOB, followed by the declaration for the actual LOB data. But the largest character and graphic variable declaration permitted in a COBOL program is 32,767 bytes. So, for LOBs greater than 32,767 bytes, DB2 will create multiple host language declarations of 32,767 or fewer bytes.

APPENDIX D

DB2 Limits

You can use this appendix as a handy reference for the various physical and structural limitations to which DB2 must conform. There are two columns, the first showing the V7 limit, the second showing the V8 limit. If the limit did not change between V7 and V8, the V8 column will be blank.

Item	V7 Limit	V8 Limit
STOGRROUP name	8 bytes	128 bytes
Volumes per STOGRROUP	133	
Database name	8 bytes	
Maximum number of databases	65,217	
Authorization ID	8 bytes	128 bytes
Tablespace name	8 bytes	
Maximum segment size	64 pages	
Largest Partitioned Table Space	16 terabytes	128 terabytes
Largest Segmented Table Space	64 gigabytes	
Largest Simple Table Space	64 gigabytes	
Table name	18 bytes	128 bytes
View name	18 bytes	128 bytes
Alias name	18 bytes	128 bytes
Synonym name	18 bytes	128 bytes
Column name	18 bytes	128 bytes
Referential constraint name	8 bytes	128 bytes
Check constraint name	18 bytes	128 bytes
Maximum length of the check constraint text	3,800 bytes	
Cursor name (except for DECLARE CURSOR WITH RETURN *)	18 bytes	128 bytes
Host identifier	64 bytes	
Location name	16 bytes	
Number of base tables per view	225	
Maximum number of columns in a table (or view)	750 **	
Index name (8 bytes recommended ***)	18 bytes	128 bytes
Columns per index	64	
Index key size (number of nullable columns ****)	255 bytes	2,000 bytes
Plan name	8 bytes	
Package name	8 bytes	
Trigger package name	8 bytes	128 bytes
Collection name	18 bytes	128 bytes
Package Version name	64 bytes	
DBRM name	8 bytes	
Schema name	8 bytes	128 bytes
Stored Procedure name	18 bytes	128 bytes
User-defined function name	18 bytes	128 bytes
Trigger name	8 bytes	128 bytes

Item	V7 Limit	V8 Limit
Maximum length of CHAR	255 bytes	
Largest VARCHAR		
(4KB pages)	4,046 bytes	
(8KB pages)	8,128 bytes	
(16KB pages)	16,320 bytes	
(32KB pages)	32,704 bytes	
Maximum length of GRAPHIC	127 DBCS characters	
Largest VARGRAPHIC		
(4KB pages)	4,046 bytes	
(8KB pages)	8,128 bytes	
(16KB pages)	16,320 bytes	
(32KB pages)	32,704 bytes	
Maximum length of BLOB	2,147,483,647 bytes	
Maximum length of CLOB	2,147,483,647 bytes	
Maximum length of DBCLOB	1,073,741,824 DBCS characters	
Largest SMALLINT	32,767	
Smallest SMALLINT	-32,768	
Largest INTEGER	2,147,483,647	
Smallest INTEGER	-2,147,483,648	
Largest DECIMAL	$10^{31} - 1$	
Smallest DECIMAL	$1 - 10^{31}$	
Largest FLOAT	7.2×10^{75}	
Smallest FLOAT	-7.2×10^{75}	
Smallest positive FLOAT	5.4×10^{-79}	
Largest negative FLOAT	-5.4×10^{79}	
Smallest DATE	0001-01-01	
Largest DATE	9999-12-31	
Smallest TIME	00.00.00	
Largest TIME	24.00.00	
Smallest TIMESTAMP	0001-01-01-00.00.00.000000	
Largest TIMESTAMP	9999-12-31-24.00.00.000000	
Physical Storage		
SMALLINT	2 bytes	
INTEGER	4 bytes	
REAL	4 bytes	
DOUBLE PRECISION	8 bytes	
DECIMAL(<i>p</i> , <i>m</i>)	(TRUNCATE(<i>p</i> /2)+1) bytes	
CHAR(<i>n</i>)	<i>n</i> bytes	
VARCHAR(<i>n</i>)	<i>n</i> + 2 bytes	
LONG VARCHAR	size of tablespace page	
GRAPHIC(<i>n</i>)	$2 * n$	
VARGRAPHIC(<i>n</i>)	$(2 * n) + 2$ bytes	
LONG VARGRAPHIC	Size of tablespace page	
DATE	4 bytes	

Item	V7 Limit	V8 Limit
TIME	3 bytes	
TIMESTAMP	10 bytes	
Maximum LOB data set size	64GB	
Row length		
(4KB pages)	4,056 bytes	
(8KB pages)	8,138 bytes	
(16KB pages)	16,330 bytes	
(32KB pages)	32,714 bytes	
Row length (with EDITPROC)		
(4KB pages)	4,046 bytes	
(8KB pages)	8,128 bytes	
(16KB pages)	16,320 bytes	
(32KB pages)	32,704 bytes	
Maximum number of rows per page		
(user tables)	255	
(DB2 Catalog & Directory)	127	
Maximum number of tables		
in a FROM clause	15	225
Maximum number of tables per SELECT/INSERT/UPDATE/DELETE	225	
Maximum number of subqueries in an SQL statement	14	
Maximum number of triggers, stored procedures, and UDFs referenced by a single SQL statement	16 nested levels	
Maximum length of SQL path	254 bytes	
Largest SQL statement	32,765 bytes	2,097,152 bytes
Columns***** per SELECT	750	
Largest string literal	255 bytes	32,704 bytes
SQL correlation ID	18 bytes	128 bytes
Predicates per WHERE clause	750	
Predicates per HAVING clause	750	
Length of columns in ORDER BY	4,000	
Length of columns in GROUP BY	4,000	
Maximum length of host and indicator variables pointed to in SQLDA	32,767 bytes	
Maximum size of a single stored procedure parm	2 gigabytes-1 for a LOB*****	
Concurrent users	2,000	
Open data sets	32,727	100,000
Largest active log data set	2GB	
Largest archive log data set	2GB	

Item	V7 Limit	V8 Limit
Maximum active log copies	2	
Maximum archive log copies	2	
Maximum active log data sets	31	93
Maximum archive log volumes	1,000	10,000
Maximum DBRM entry size	131,072 bytes	
<i>* The maximum size of a cursor declared WITH RETURN is 30 bytes.</i>		
<i>** If the table is a dependent, it can contain a maximum of 749 columns. The value (749 or 750) depends on the complexity of the CREATE VIEW statement.</i>		
<i>*** If the index name is longer than 8 bytes, DB2 derives an index space name using the index name. An index space name must be unique in the given database. The index space name that DB2 generates for index names of nine characters or more may be hard to track when you're performing DASD management and object monitoring. It also is helpful to have the index space names be the same when comparing files from different environments; for example, comparing QA to production.</i>		
<i>**** The maximum length of the key for a partitioning index remains 255 even for DB2 V8. For V7 and V8, both partitioning and nonpartitioning indexes, you must subtract 1 for each nullable column in the index to determine the total maximum length of the columns that can be assigned to the index. For V8 padded indexes, you must also subtract 2 times the number of varying-length columns in the key.</i>		
<i>***** The maximum is for all items in the SELECT list, not just columns. For example, expressions and constants can be included in the SELECT list.</i>		
<i>***** The maximum number of bytes passed for LOB data is subject to the capabilities of the application programming language and environment being used.</i>		

IN THIS APPENDIX

- The DB2 Family
- Packaging and Naming Issues

APPENDIX E

DB2 on Other Platforms

Although DB2 began its life on the mainframe, the advent of client/server technology and the success of competing RDBMS products (such as SQL Server and Oracle) caused IBM to create versions of DB2 for additional platforms. In short, DB2 is no longer just a mainframe product.

The DB2 Family

Versions of DB2 exist for a large array of platforms, of which the mainframe (z/OS and OS/390) is only one (see Table E.1). These products are all collectively referred to by IBM as the *DB2 Family*. Individually, each DBMS is referred to as *DB2*, or *DB2 Universal Database Server*. The proper way to refer to any individual offering in the DB2 family is *DB2 for (operating system)* (for example, DB2 for z/OS or DB2 for Windows). If the version of the DB2 product has been extended with object/relational capabilities, it is referred to as *DB2 UDB for (operating system)*, where the UDB stands for Universal Database. Version 6 was the first version of DB2 for z/OS that earned the UDB moniker.

Many shops implement applications on several different platforms and interconnect them using client/server development methods.

TABLE E.1 The DB2 Family of Products

Platform	Operating System	AKA (Old Name)
iSeries (AS/400)	OS/400	SQL/400
Mainframe	MVS, OS/390, z/OS	DB2
Mainframe	VM	SQL/DS
Mainframe	VSE	SQL/DS
Intel	Windows NT/95/98	---
	Linux	---
	SCO UNIXware	---
IBM pSeries	AIX	DB2/6000
Hewlett Packard	HP-UX	---
Sun	Solaris	---
PDA	Palm Computing	---
	PocketPC	Windows CE

NOTE

The PDA version of DB2 is referred to as *DB2 Everyplace*. In addition to Palm and Windows PocketPC, DB2 Everyplace can run in the following environments: Symbian, embedded Linux, QNX Neutrino, Microsoft Win32 and Linux.

Different Code Bases

There are four distinct code bases for the products under the DB2 brand. The mainframe has its own code base, as does the iSeries, and VSE/VM. The fourth code base is for Linux, Unix, and Windows (LUW) platforms—and all of the other DB2 offerings originate from this code base.

Having a separate code base means that each of these DB2 “products” was developed independently—at least in the beginning. So, for example, the process used by DB2 for z/OS to optimize SQL differs from the process used by DB2 for Linux. Hopefully, though, the result is similar—an efficient SQL statement.

So, there will be some major differences between the DB2s.

Some Major Differences

It is obvious that the different DB2 products are not “plug and play” commodities simply because they all share the name *DB2*. There are some big differences among these products in their current releases. The biggest differences are relatively easy to detect and include the following:

- Differences imposed due to operating system constraints (OS/400 versus OS/390 versus AIX)
- Back-level compatibility issues

- Workstation orientation differences such as GUI interfaces and drag-and-drop menus
- Subsystem-centric implementation (z/OS and OS/390) versus database-centric implementation (workstation)

Most of these differences are minor and easy to handle. Indeed, IBM has slowly but surely been making these disparate implementations of DB2 more and more alike with each new release and version. However, there are some “gotchas” lurking under the covers that might be more difficult to find.

Of the basic differences mentioned earlier, the only one that might not be obvious is the focus of the DBMS implementation. DB2 for LUW is database-centric. This implies that each new database carries its own system catalog with it. Additionally, it is not possible to simply access tables across different databases; distributed access is required.

On z/OS, DB2 is subsystem-centric. A single system catalog spans databases. Each subsystem has a unique identification, and you can create multiple databases within it. Distributed requests are not required to access databases within the same subsystem (or, indeed, across multiple subsystems in a data-sharing environment).

Directories

Another concept that is different at the workstation level is that of a directory. The DB2 for OS/390 Directory houses DBMS system-related information regarding DBD structure, skeleton plan and skeleton package tables, RBA log ranges, and utility control data. The information cannot be updated by the user but is managed and controlled by DB2.

At the workstation level, a directory is another matter altogether. For example, the directory structure used by DB2 for LUW controls the overall environment. The directories used by DB2 for LUW are as follows:

- The *System Database Directory* identifies the databases that can be accessed from the workstation and contains an entry for each local and remote one. Each database entry contains the database name, alias, entry type, and location.
- One *Volume Database Directory* is allocated per disk drive that contains a workstation database. Each entry identifies the location of a specific database on the drive.
- The *Workstation Directory* is used to make a connection to a remote database server. It is used in conjunction with the Database Connection Services Directory to make a connection to a remote host server.
- The *Database Connection Services Directory* is used by DB2 Connect to make a connection to a remote host server.

Not only is it possible for the user to update these directories, it is required. The workstation directories define the environment of DB2 for LUW. Without the proper information recorded in these directories, DB2 might not function in the desired manner. The information in these directories is somewhat analogous to DB2 for OS/390 DSNZPARMs and SYSDDF.

Database Structures

Not all the objects available to DB2 for OS/390 users are supported at the workstation level. For example, hardware-specific DB2 objects such as tablespaces and storage groups are not available for DB2 on other platforms, at least as we are used to dealing with them. Partitioning and segmenting as it is done on the OS/390 flavor of DB2 is not possible. However, DB2 for common servers does provide a feature known as a segmented table. But this is not the same concept as a DB2 for OS/390 segmented tablespace. Common server segmented tables are used to span volumes, enabling DB2 to get around the 2 gigabyte file size limitations under AIX.

The file structure used for databases differs from platform to platform. For example, DB2 for OS/390 uses VSAM Linear Data Sets (LDS) or Entry Sequenced Data Sets (ESDS). A database deployed on DB2 for common servers uses two files for table data: one for normal data and a second to store long fields. These workstation files are flat files, not VSAM files.

Although tables are basically the same for all of the DB2 environments, not all of the DDL options are provided in all of the environments. For example, DB2 for OS/390 does not support triggers, and DB2 for common servers does not allow VALIDPROCS, FIELDPROCS, and EDITPROCS.

Optimizer Differences

One of the most significant benefits of relational databases is that they provide built-in optimization. The DB2 for OS/390 optimizer is well-known to mainframe DB2 users, but how similar are the other DB2 optimizers?

The DB2 for common servers product uses the latest and greatest optimization technology from IBM's Almaden labs: the Starburst optimizer. Starburst is a database optimization research project that has been covered quite extensively in the academic press. Although some Starburst technology will find its way to DB2 for OS/390, the DB2 for OS/390 optimizer will never be completely replaced by Starburst technology. The DB2 for OS/390 optimizer has been finely tuned for its environment over the course of more than a decade.

Another interesting tidbit is that DB2 for iSeries provides an access method for programmers in which they can bypass the relational engine. This is not encouraged, but it is available.

Other Differences

Other differences exist between the different implementations of DB2. Some of these are caused by the different release cycles IBM has created for the differing platforms. For example, DB2 for Linux, Unix, and Windows has supported recursive SQL for several releases, but DB2 for z/OS just recently added recursive SQL support in Version 8.

The bottom line is that you need to be aware that there are differences between the DB2s on different platforms. Whenever you use a specific implementation of DB2, you need to be aware of the features it supports that other DB2 platforms do not, as well as the features it does not support that other DB2 platforms do support.

Packaging and Naming Issues

The actual name of the DB2 edition can be tricky to master on non-mainframe platforms. On the mainframe you just say “I want DB2,” and that is what you get. Well, almost. You also have to decide whether you want IBM’s utilities or not, too. But things are more difficult in the LUW world. The following packages are all available for DB2 on Linux, Unix, and Windows.

DB2 Workgroup Server Edition (WSE) is a multi-user, single host, Web-enabled database with included Java support, but without support for inter- or intra-partition parallelism. WSE fits most closely into the departmental DBMS architecture. It should be used for small systems with a limited number of users. WSE has a registration limit of a maximum of four processors.

DB2 Workgroup Server Unlimited Edition (WSUE) has the same functionality, and the same registration limit of a maximum four processors, but a different pricing model. So WSUE seems to fall somewhere between a departmental and an enterprise DBMS architecture—for larger numbers of users than WSE.

DB2 UDB Enterprise Server Edition (ESE) is the highest level of DB2 database version with intra-partition parallelism support (the database engine can process SQL statement segments in parallel), and inter-partition parallelism support (process a query in parallel across all of the nodes). ESE has Partitioning and Clustering options as additional add-on features. So, this is the enterprise DB2. This is what used to be known as EEE, sort of.

DB2 UDB Personal Edition (PE) is a single-user database engine ideal for deployment to PC based users. PE is IBM’s personal architecture DBMS offering.

DB2 UDB Developer’s Edition (DE) is a low cost package for a single application developer to design, build, and prototype applications for deployment on any of the DB2 client or server platforms. Here is another option for your developers. To save costs you can license DE instead of ESE, WSE, or WSUE, but you cannot use DE for production work.

DB2 UDB Personal Developer’s Edition (PDE) enables a developer to design and build single user desktop applications. Similar to DE, PDE is a lower cost development option for single user application development.

Then there is **DB2 Express**. It was announced in the first quarter of 2003 and IBM touts it as “a specially tailored database offering for the worldwide small and medium business (SMB) market, with focus on enterprises of size 100 to 1000 employees.” This “edition” is targeted at IBM’s partners, encouraging them to build applications on top of DB2 Express targeting the SMB market.

The list of DB2s does not stop here. IBM has developed specialty editions of DB2 to handle heterogeneous, federated data and data that is not traditionally stored in databases.

Federation and Information Integration

No matter how entrenched DB2 might be in your organization, you’ll likely have and use other DBMS products, too. Homogeneous IT organizations are extremely rare. Given that

fact, how do businesses handle the need to access data from multiple disparate data sources? IBM attacks this problem through federation.

Federation enables businesses to access and integrate data from multiple locations as if it were stored in a single location. Federation enables customers to abstract a common data model across data and content sources and to access and manipulate them as though they were a single source.

IBM's **DB2 Information Integrator** offers the ability to integrate data from multiple sources of different kinds of information. The product enables access to data from hierarchical DBMSs such as IMS, other relational products such as Microsoft SQL Server, and other sources. When it's returned to the application, the data appears as if it came from DB2.

Content and Records Management

As businesses, government agencies, and other organizations continue to store more data of multiple types, applications will need to manage and unite all kinds of information—structured data plus emails, phone recordings, faxes, video, and other information sources—to solve business problems. RDBMSs traditionally focus on managing structured data, data that can be stored in rows and columns in a table. DB2 extenders expand the kinds of information it's possible to store; however, sometimes more in-depth manipulation and management of this unstructured, complex data is needed.

DB2 Content Manager provides support for two kinds of content management: media asset management and enterprise content management. Media asset management is the storage and management of collections of large multimedia objects, such as large collections of X-rays for a hospital, video and film content for movie studios, and scans of art collections for museums. Enterprise content management involves the storage and management of large collections of smaller multimedia objects—often scanned check images for banks or scanned invoices, bills, or similar data for many kinds of businesses.

DB2 Information Integrator for Content provides a programming layer above DB2 Content Manager that facilitates easy access to many kinds of information from a single interface.

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 stand-alone 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 cursor 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, LASTNAME
FROM      DSN8710.EMP
ORDER BY LASTNAME;

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,
 FIRSTNAME  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 DSNZPARMS 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.

APPENDIX G

DB2 Version 8 Overview

This appendix provides you with a concise overview of the *major* features and functions of DB2 Version 8. Keep in mind, though, that this short appendix is merely an overview of the great features you can expect in DB2 V8. In-depth information on particular DB2 V8 topics is contained throughout the text of this book.

For an exhaustive overview of DB2 V8, read the IBM manual “DB2 UDB for z/OS Version 8 Technical Preview” (SG24-6871). This manual can be downloaded for free from the IBM Web site.

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

- Architecture
- Database Administration
- Programming and Development
- Migration to DB2 V8

Architecture

Let’s start by reviewing the significant changes to the architecture of DB2 and related requirements. One of the biggest impacts of V8 will be the requirement to be running a zSeries machine and z/OS v1.3—DB2 V8 will not support old hardware nor will it support OS/390. Additionally, DB2 customers must migrate to V7 before converting to V8. There will be no IBM-supported capability to jump from V6 (or an older version) directly to V8 without first migrating to V7.

Owing to these architectural requirements, DB2 will have the ability to support large virtual memory. This next version of DB2 will be able to surmount the limitation of 2GB real storage that was imposed due to S/390’s 31-bit addressing. Theoretically, with 64-bit addressing DB2 could have up to 16

IN THIS APPENDIX

- Architecture
- Database Administration
- Programming and Development
- Migration to DB2 V8

exabytes of virtual storage addressability to be used by a single DB2 address space. Now there is some room for growth!

Broader usage of Unicode is another architectural highlight of DB2 V8. V7 delivered support for Unicode-encoded data, but V8 forces its use. If you do not use Unicode today, you will when you move to V8. This is so because many of the table spaces in the DB2 system catalog will be implemented using Unicode. In fact, the DB2 catalog has some other dramatic changes coming under V8—including some table spaces with larger page sizes and long names.

Actually, support of long DB2 object names is another significant architectural change in V8. DB2 V8 significantly increases the maximum length of most DB2 object names. For example, instead of being limited to 18 byte table names, you will be able to use up to 128 bytes to name your DB2 tables; the same limit applies to most DB2 objects and special registers including views, aliases, indexes, collections, schemas, triggers, and distinct types. The limit for columns is 30 bytes, a table space is still 8 bytes, and packages are still 8 bytes, unless it is a trigger package, which can be 128 bytes. This brings a lot of flexibility, but also a lot of reworking of the DB2 catalog tables.

One such reworking requires the use of table spaces with 8K, 16K, and 32K page sizes. Therefore, the system catalog in DB2 V8 will require use of the BP8K0, BP16K0, and BP32K buffer pools.

Database Administration

As with each new version, DB2 V8 offers new functionality that helps DBAs administer and manage their databases and subsystems. This release contains many enhancements to the DB2 objects that DBAs must manage including sequence objects, variable length index keys, expanded partitions, new types of partitioned indexes, new partition management, and materialized query tables (also known as automated summary tables). Also, index keys can comprise up to 2000 bytes—so more data can be indexed using a single index. Each of these features delivers more functionality but also presents implementation and maintenance challenges.

Sequences

Identity columns, added during the DB2 V6 refresh, can be useful, but there are numerous problems involved when trying to actually use them. The biggest problems come about when data must be loaded into these tables because you cannot control the identity values assigned.

SEQUENCE objects resolve most of the problems with identity columns. A SEQUENCE object is a separate database object that generates sequential numbers. When a SEQUENCE object is created, it can be used by applications to “grab” a next sequential value for use in a table.

Sequences are efficient and can be used by many users at the same time without causing performance problems. Multiple users can concurrently and efficiently access SEQUENCE objects because DB2 does not wait for a transaction to COMMIT before allowing the

sequence to be incremented again by another transaction. Sample DDL for creating a SEQUENCE object follows:

```
CREATE SEQUENCE ACTNO_SEQ
  AS SMALLINT
  START WITH 1
  INCREMENT BY 1
  NOMAXVALUE
  NOCYCLE
  CACHE 10;
```

This SEQUENCE object can be used to generate sequential values in SQL statements, using sequence expressions. DB2 V8 supports two sequence expressions:

- NEXT VALUE FOR to automatically generate the next value
- PREV VALUE FOR to return the last generated value

The following sample SQL uses a sequence expression to generate the next sequential value and uses that value in an INSERT statement:

```
INSERT INTO DSN8810.ACT
  (ACTNO, ACTKWD, ACTDESC)
VALUES
  (NEXT VALUE FOR ACTNO_SEQ, 'TEST', 'Test activity');
```

Sequence expressions are not limited to INSERT statements, but can be used in UPDATE and SELECT statements, too.

Partitioning

Major changes to the way DB2 partitions data are introduced in V8. First of all, the partitioning limit keys now are defined in the table, instead of a partitioning index. In fact, no partitioning index is required.

Furthermore, clustering and partitioning have been separated, so you can cluster on one group of columns and partition on another. Also, DB2 V8 expands partitioning support to allow for much-needed data growth. You can define up to 4,096 partitions per partitioned table space with DB2 V8.

With online schema evolution, discussed later, making changes to partitioned table spaces is much easier. And a new type of partitioned index is introduced.

Data Partitioned Secondary Indexes

One of the biggest problems DBAs face when they are managing large partitioned DB2 table spaces is contending with non-partitioned indexes. DB2 V8 introduces data partitioned secondary indexes, or DPSIs, to help resolve these problems.

A DPSI is basically a partitioned NPI. Even though the index key for a DPSI is not the partitioning key, DB2 will manage the partitioning in the index such that entries are maintained in the same partition where the data is found in the partitioned table space. In

other words, a DPSI will be partitioned by the same key ranges as the table, whereas an NPI is not partitioned at all.

As of V8 another term for NPI is NPSI, or non-partitioned secondary index.

So, with a DPSI the index will be partitioned based on the data rows. The number of parts in the index will be equal to the number of parts in the table space. This helps with partition-based utility processing, because now DB2 utilities can process the DPSI partition at the same time it processes the table partition.

But you will not want to change every NPI to a DPSI once you migrate to V8. Changing an NPI to a DPSI will likely cause queries to perform worse than before. This is so because each partition of a DPSI has its own index tree structure. So queries may have to examine multiple partitions of the DPSI as opposed to the single NPI it previously used. This can degrade performance.

Online Schema Evolution

Another useful administration feature of DB2 V8 is known as Schema Evolution. Today, there are many types of DB2 changes that require the DBA to DROP and then re-CREATE the object in order to implement the change. Schema evolution enables the DBA to make more types of changes to database objects using native DB2 features. For example, DBAs will be able to add and rotate partitions of partitioned table spaces and to expand the length of numeric and character columns using the ALTER statement. Basically, schema evolution provides more support for a variety of changes to be made directly using ALTER statements.

An in-depth discussion of online schema evolution is provided in Chapter 7, “Database Change Management and Schema Evolution.”

Multi-Level Security

The new security features are interesting, too! DB2 V8 introduces multi-level security. With multilevel security (MLS) in DB2 V8 it becomes possible to support applications that need a more granular security scheme. For example, you might want to set up an authorization scenario such that employees can see their own data but no one else's. To complicate matters somewhat, you might also want each employee's immediate manager to be able to see his payroll information as well as all of his employee's data, and so on up through the org chart. Setting up such a security scheme is next to impossible with current DB2 versions, but it is straightforward using row level security in DB2 V8.

DB2 V8 supports row-level security in conjunction with a security management product (like RACF). To activate this authorization mechanism, you will need to add a specially named column to act as the security label. The security label column is matched with the multilevel security hierarchy in the security manager. You can set up security hierarchies to be as simple, or as complex, as you need. To support MLS hierarchies, DB2 V8 requires several new RACF access control functions that are not available prior to V1R5 of z/OS.

When row-level security is implemented, every user must be identified to RACF (or another security server with equivalent functionality) with a valid SECLABEL. Row-level

security is then implemented by matching the SECLABEL of the data to the SECLABEL of the user. But, of course, there is additional detail that is needed to implement user row-level authorization properly in DB2. This detail can be found in Chapter 10, “DB2 Security and Authorization.”

Padded Variable Length Indexes

Prior to Version 8, when indexing on a variable column, DB2 automatically pads the variable column out to its maximum size. So, for example, creating an index on a column defined as `VARCHAR(50)` will cause the index key to be padded out to the full 50 bytes. Padding very large variable columns can create a very large index with a lot of wasted space.

DB2 V8 offers the capability to direct DB2 whether variable columns in an index should be padded or not using a new keyword in `CREATE INDEX`: `PADDED` or `NOT PADDED`. The specification is made at the index level—so every variable column in the index will be either padded or not padded.

When `PADDED` is specified, DB2 will create the index just as it did prior to V8—by padding all variable columns to their maximum size. When `NOT PADDED` is specified, DB2 will treat the columns as variable and you will be able to obtain index-only access because the length is stored in the index key.

Support for Greater Log Volume

The maximum number of archive log volumes recorded in the BSDS expands to 10,000 volumes per log copy from the previous limit of 1,000 volumes. The maximum number of active log data sets is also increased from 31 per log copy to 93.

To obtain this increased number of log data sets you must re-size your BSDS. This is accomplished by running `DSNJCNVB`. DB2 V8 must be running in New Function Mode before you can modify the BSDS.

NOTE

Although BSDS conversion is optional, it is wise to convert it to take advantage of the expanded log volume support.

Additionally, the maximum size of each log data set (both active and archive) can be up to 4 MB minus 1 CI. Although this increase is available in the base code for DB2 V8, it is also available to DB2 V6 and V7 via APAR `PQ48126`.

Additional V8 DBA Improvements

This section covered the highlights of this version’s DBA improvements. But there are others. For example, you can create MQTs, or materialized query tables, to improve the performance of data warehousing queries. MQTs are essentially views where the data has been physically stored instead of virtually accessed.

Other improvements include

- Long object names, permitting DBAs to create standards allowing greater descriptive names
- The ability to specify the actual CI size of the underlying VSAM data set for DB2 table spaces to synchronize it with the DB2 page size (8K, 16K, or 32K)
- Two new utilities (`BACKUP SYSTEM` and `RESTORE SYSTEM`) for managing system-level, point-in-time backup and recovery
- Enhanced `RUNSTATS` capability for collecting additional distribution statistics, thereby enhancing query optimization
- Support for delimited `LOAD` and `UNLOAD` data sets
- Data sharing enhancements to provide CF lock propagation reduction, a reduction in overhead for data sharing workloads, batched updates for index page splits, improved LPL recovery, and improvements to data sharing-related commands
- Additional `DSNZPARMs` can be changed online (though it is still not possible to change **all** `DSNZPARMs` online); a complete list of what can and cannot be changed online is included in the IBM DB2 Installation Guide manual
- Improvements to identity column management—for example, changing the `GENERATED` parameter is now permitted

Programming and Development

Numerous SQL and programming features are being added to DB2 V8 that will make the job of programming both easier, but at the same time, more complex. This might sound like a paradox, but it is true. Great new features will make programming simpler once they are learned, but it will take time and effort to train the legions of DB2 developers on this new functionality, and when and how best to use it.

Common Table Expressions and Recursion

One big improvement in V8 is the ability to code *common table expressions (CTEs)*. A common table expression can be thought of as a named temporary table within a SQL statement that is retained for the duration of a SQL statement. There can be many CTEs in a single SQL statement but each must have a unique name and be defined only once. A CTE is defined at the beginning of a query using the `WITH` clause.

CTEs are an important new feature of DB2 for several reasons. First, in some situations they can be used to reduce the number of views that are needed. Instead of creating a view, you can code a CTE right into your query. But second, and more importantly, CTEs enable recursive SQL.

A recursive query is one that refers to itself. I think the best way to quickly grasp the concept of recursion is to think about a mirror that is reflected into another mirror and when you look into it you get never-ending reflections of yourself. This is recursion in action.

Recursive SQL can be very elegant and efficient. However, because of the difficulty developers can have understanding recursion, it is sometimes thought of as “too inefficient to use frequently.” But, if you have a business need to walk or explode hierarchies in DB2, recursive SQL will likely be your most efficient option. Recursion is covered in more depth in Chapter 2 of this book.

Architecture Changes Impacting Application Programming

V8 offers significant changes to the SQL system limits. First, as we have already mentioned, DB2 will now offer long name support for database objects. But it does not stop there. DB2 V8 expands the maximum length of SQL statements to support up to 2 megabytes. This is a major change that permits much more complex SQL statements to be written, optimized, and run within DB2. Additionally, V8 increases the length of literals and predicates to 32K and will support joining up to 255 tables in a single SQL statement. This last one has been promised before, but is finally delivered properly in V8.

Also, as noted in the initial architecture section, 64-bit virtual addressing will greatly increase the amount of memory available to DB2. And IBM is making major enhancements to the internal SQL control block structures, so that DB2 will use memory more efficiently. So more memory, used more efficiently, should translate into the more efficient execution of DB2 SQL.

Java and XML Improvements

For Java programmers DB2 V8 offers expanded functionality in the form of support for both Type 2 and Type 4 Java drivers. Both will be updated to support the JDBC/SQLJ 3.0 standard which brings enhanced support for things such as SAVEPOINTS and WITH HOLD cursors, as well as improvements to connection pooling, and a long list of other expanded features.

DB2 V8 pushes more XML support into the DB2 engine. This includes support for some built-in XML publishing functions such as XMLELEMENT and XML2CLOB (among others).

Additional V8 Programming Improvements

But there are many more application-related improvements in DB2 V8 than just CTEs and recursion. For example, DB2 V8 removes one of the biggest SQL performance impediments of all-time by handling most unlike data types in Stage 1. Previously, if the data type and length of the columns and variables did not match exactly, the predicate was evaluated at Stage 2. DB2 V8 will compare unlike data types in Stage 1 as long as the data types are compatible (that is, number to number, or character to character, and so on).

And there are more application enhancements worth noting in DB2 V8. Consider all of these new features:

- A new statement, GET DIAGNOSTICS, is added that improves the ability to get diagnostic information.
- SEQUENCE objects and sequence expressions.
- New MQSeries functions to read and receive from queues.

- Dynamic scrollable cursors that no longer require temporary tables to implement.
- Scalar fullselect, which means that a SELECT statement that returns a single can be used wherever an expression is allowed.
- More than one DISTINCT clause can now be specified per SQL statement.
- The ability to mix EBCDIC, ASCII, and Unicode columns in a single SQL statement.
- Qualified column names on the SET clause of INSERT and UPDATE statements.
- Grouping expressions can be used in search conditions in HAVING clauses, in the SELECT clause, and in sort key expressions of an ORDER BY clause.
- The ability to SELECT from an INSERT statement.
- Multi-row FETCH and INSERT statements where more than one row can be fetched or inserted by a single statement.

And, as with every previous new DB2 version, IBM is making significant enhancements to improve application performance. DB2 V8 optimization enhancements are scheduled to include sophisticated query rewrite capabilities to support materialized query tables, sparse indexing to improve star join performance, support for parallel sort, and better support for queries with data type and length mismatches which would have caused less efficient access paths in previous releases.

Migration to DB2 V8

When it comes time to manage the migration of your DB2 subsystems to Version 8 you will need to better understand the significant differences for V8 migration than for your previous DB2 migration strategies. The biggest difference is the introduction of three distinct modes that dictate how DB2 operates and the functionality that you will have available to you.

But before we discuss these three new DB2 modes, let's quickly examine the basics of DB2 version migration. Typically, when you decide to begin using a new version of DB2 you will migrate your test subsystems to the new version. Over time and after testing, when you decide that everything looks fine, you “throw the switch” and migrate your production subsystems. When a subsystem is running on the new release (whether test or production) all the functionality of the new version is available to all DB2 users. Of course, for fallback purposes, many shops try to discourage the immediate use of new functionality, preferring to be sure the new release is stable. But, prior to V8, there was no mechanism to support such a phased roll-in of new functionality.

You will be able to exploit the three modes of DB2 to help manage how new functionality is used as you migrate to V8. The three modes are *compatibility mode (CM)*, *enabling new function mode (ENFM)*, and *new functionality mode (NFM)*.

As you begin the migration process DB2 V8 will begin in compatibility mode. No new functionality is available at this stage. A DB2 V8 subsystem in compatibility mode is ideal for verifying functionality of existing applications and processes to ensure that they

function as they did in Version 7. After this verification is complete, there is no real need to remain in compatibility mode any longer.

The next phase of the migration process moves DB2 V8 into enabling new function mode. Job DSNTIJNE is run to begin the process of enabling new functionality. At this stage conversion of critical subsystem components has begun, but as long as you remain in this mode, most new functionality is not available to users. Certain DB2 system catalog changes are made during this mode such as the movement of several (not all) table spaces to Unicode, the extension of many existing columns to support long names, and alteration of certain catalog indexes to be NOT PADDED (because VARCHAR is used for long name columns). Additionally, keep in mind that fallback to CM mode or to V7 is **not** permitted once you have entered ENFM mode.

You can remain in ENFM mode for as long as you need to complete the task. IBM supplies the DSNTIJNH job that can be run to halt the enabling new function job. In this way you can stage enabling new functionality over time. To pick up where you left off, simply run the DSNTIJNE job again and it will figure out where it was halted and start running again. This is a nice feature if you only have a limited window where you can make changes to the DB2 catalog because it permits phased implementation of the required changes.

The final stage of migrating to DB2 V8 is new functionality mode. Job DSNTIJNF is run to move into new function mode. When your subsystem is moved into this mode all of the new V8 functionality is available and you have successfully migrated to DB2 Version 8. In NF mode, all of the necessary DB2 system catalog changes are complete including the addition of new tables and columns, and any needed modification of existing columns. Additionally, several table spaces will have grown to be too large for 4K pages causing the DB2 catalog to require 8K, 16K, and 32K page sizes and buffer pools for the first time.

Also, keep in mind the following rules as you migrate to DB2 V8 and progress from CM mode to EN mode to NF mode:

- You must be at DB2 V7 in order to migrate to DB2 V8; there is no migration to V8 from any previous version or release of DB2.
- You will need to apply the proper level of maintenance to DB2 V7, before migrating to V8, in order to be able to fall back to V7 from CM mode. If you have not applied the fallback SPE, your DB2 V8 migration will fail and you will have to start over by first applying the fallback SPE to V7 and then proceeding with your migration.
- Although you cannot fall back to V7 after you move to NFM mode, you can fall back to ENFM mode. This can be useful to curtail usage of V8 functionality, if you suspect that it is causing problems.
- The migration process will change any user-defined indexes that you have built on your DB2 catalog tables, but these indexes will not be changed to NOT PADDED. If these indexes contain any column that refers to a long name, then your indexes will become very large until you alter them to be NOT PADDED.
- Be sure to migrate any existing type 1 indexes to type 2. DB2 V8 will fail if any type 1 indexes are found in the catalog.

CAUTION

When you move to NF mode, DB2 will create DBRMs in Unicode. This can complicate migrations from a V8 development system to a V7 production system.

The migration process for DB2 V8 is quite different from any previous DB2 release migration. Be sure to study the DB2 manuals to understand all of the nuances of each mode before beginning the migration of your DB2 subsystems to V8.

APPENDIX H

Reorganizing the DB2 Catalog

IN THIS APPENDIX

- When Should the DB2 Catalog and Directory Be Reorganized?

Prior to DB2 V4, it was not possible to reorganize the table spaces in the DB2 Catalog and DB2 Directory because of the internal hash and link structures built into these databases. But for DB2 V4 and later releases, you can expediently reorganize the DB2 catalog and DB2 directory in a systematic manner using the REORG utility.

The DB2 catalog is the central repository for DB2 object and user metadata. DB2 is constantly referring to that metadata as it processes applications and queries. The physical condition of the table spaces and indexes that comprise the DB2 catalog is therefore a major component in overall DB2 subsystem performance.

Likewise, the DB2 directory contains internal control structures such as DBDs and skeleton cursor tables that can be accessed only by DB2 itself. The information in the DB2 directory is critical for database access, utility processing, plan and package execution, and logging. Efficient access to this information is quite critical.

Prior to DB2 V4, the only option for any type of “reorganization” activity was to run the RECOVER INDEX utility on DB2 catalog indexes. This rebuilt the indexes, but had no impact on the underlying data housed in the actual physical table space. Catalog reorganization is permitted on table spaces and indexes in the DB2 catalog database (DSNDB06) and on specific table spaces (SCT02, SPT01, and DBD01) in the DB2 directory database (DSNDB01). As of DB2 V6, of course, the REBUILD INDEX utility can be run to rebuild DB2 Catalog and Directory indexes.

When Should the DB2 Catalog and Directory Be Reorganized?

To determine when to reorganize the system catalog, DBAs can use most of the same basic indicators used to determine whether application table spaces should be reorganized. Although it always has been a wise course of action to execute RUNSTATS on the DB2 Catalog table spaces, it becomes even more important now that these table spaces can be reorganized. These statistics can be analyzed to determine when a REORG should be run. When RUNSTATS is run for a catalog table space, the statistics about that system catalog table space are gathered and then stored in the DB2 Catalog tables themselves.

In general, the following indicators and situations should be reviewed when determining when to reorganize your system catalog table spaces and indexes:

- An increase in the value of the near- and far-off position indicators (NEAROFFPOSF and FAROFFPOSF in SYSINDEXPART)
- An increase in the value of the near and far indirect reference indicators (NEARINDREF and FARINDREF in SYSTABLEPART)
- A decrease in cluster ratio (CLUSTERRATIOF in SYSINDEXES)
- An increase in leaf distance (LEAFDIST in SYSINDEXPART)
- When the DB2 Catalog and Directory data sets are not using a significant portion of their allocated disk space (PRIQTY)
- When the DB2 catalog and directory data sets contain a large number of secondary extents

For the SYSDBASE, SYSVIEWS, and SYSPLAN catalog table spaces, the value for the FAROFFPOSF and NEAROFFPOSF columns of SYSINDEXPART can be higher than for other table spaces before they need to be reorganized. Additionally, you can use REORG when it is necessary to move the DB2 Catalog and Directory to a different disk device.

Synchronizing System Catalog Reorganization

It is a more difficult prospect to determine when the DB2 Directory table spaces should be reorganized. The RUNSTATS utility does not maintain statistics for these “table spaces” like it can for the DB2 Catalog. However, it is possible to base the reorganization of the DB2 Directory table spaces on the reorganization schedule of the DB2 Catalog table spaces. In fact, in certain situations, it is imperative that specific DB2 Directory table spaces are reorganized when a “companion” DB2 Catalog table space is reorganized. The chart in Table H.1 provides information on keeping the DB2 catalog and DB2 directory table spaces “in sync.”

TABLE H.1 DB2 Directory Reorganization Indicators

When You REORG...	Be Sure to Also REORG...
DSNDB06.SYSDBASE	DSNDB01.DBD01
DSNDB06.SYSPKAGE	DSNDB01.SPT01
DSNDB06.SYSPLAN	DSNDB01.SCT02

These table spaces are logically related. DB2 requires that you reorganize them at the same time to keep them synchronized.

DB2 Catalog Reorganization Details

There are 20 DB2 Catalog table spaces and six DB2 Directory table spaces (refer to Tables H.2 and H.3). DB2 has different rules for different sets of these table spaces. There are three groupings of table spaces:

- Cannot be reorganized at all
- Can be reorganized using normal REORG procedures
- Can be reorganized using special REORG procedures

TABLE H.2 DB2 Catalog Table Spaces (DSNDB06)

Table Space	Definition
SYSCOPY	Contains image copy information (2 tables)
SYSDBASE	Contains database object information (13 tables)
SYSDBAUT	Contains database and database authority information (2 tables)
SYSDDF	Contains information about distributed DB2 connections (8 tables)
SYSGPAUT	Contains resource authority information (1 table)
SYSGROUP	Contains storage group information (2 tables)
V7 SYSGRTNS	Contains information about DB2 routines, such as functions and procedures (2 tables)
V7 SYSHIST	Contains historical statistics (8 tables)
V7 SYSJAUXA	LOB table space for Java JAR BLOB data (1 table)
V7 SYSJAUXB	LOB table space for Java source CLOB data (1 table)
V7 SYSJAVA	Contains information about Java programs (3 tables)
SYSOBJ	Contains object/relational and routine information (10 tables)
SYSPKAGE	Contains package and stored procedure information (8 tables)
SYSPLAN	Contains plan information (5 tables)
V7 SYSSEQ	Contains sequence information (pre-V8) (1 table)
V8 SYSSEQ2	Contains sequence information (V8) (2 tables)
SYSSTATS	Contains optimization statistics (6 tables)
SYSSTR	Contains translation and check constraint information (4 tables)
SYSUSER	Contains user authority information (1 table)
SYSVIEWS	Contains view information (4 tables)

TABLE H.3 DB2 Directory Table Spaces (DSNDB01)

Table Space	Definition
DBD01	Contains database descriptor information (one table)
SCT01	Contains skeleton cursor table information (one table)
SPT02	Contains skeleton package table information (one table)
SYSLGRNX	Contains recovery log range information (one table)
SYSUTILX	Contains utility processing information (one table)

There are only two table spaces in the first grouping of table spaces that cannot be reorganized at all: DSNDB01.SYSUTILX and DSNDB01.SYSLGRNX. Do not attempt to reorganize these table spaces as DB2 will not permit it.

The second grouping of table spaces must be processed differently than other table spaces:

- DSNDB06.SYSDBASE
- DSNDB06.SYSDBAUT
- DSNDB06.SYSGROUP
- DSNDB06.SYSPLAN
- DSNDB06.SYSVIEWS
- DSNDB01.DBD01

These six table spaces require special “handling and care.” Because they have a different internal configuration than most other table spaces, a different calculation is required for the size of the unload data set (SYSREC) used during the REORG utility. These table spaces contain internal links. Links are internal pointers that tie the information in their tables together hierarchically. A link can be thought of as a type of parent-child relationship in which, due to these links, the BUILD and SORT phases of the REORG utility are not executed.

The WORKDDN, SORTDATA, SORTDEVT, and SORTNUM options are ignored when reorganizing these table spaces. Also, the REORG utility cannot be restarted from the last checkpoint when used against these six table spaces. Instead, it must be restarted from the beginning of the phase. Finally, as mentioned before, a different set of steps must be executed during reorganization for these table spaces.

All other DB2 Catalog and DB2 Directory table spaces can be reorganized like any other DB2 table space. Keep in mind that the LOB table spaces in the DB2 Catalog are under the same restrictions as any other LOB table space regarding DB2 utilities.

Steps to REORG the Six “Special” Table Spaces

The following steps should be used when reorganizing the six “special” table spaces:

1. Calculate the size of the unload data set (SYSREC).

The SYSREC data set for the “special” table spaces has a different format than the other table spaces. This causes a special calculation to be required to determine its size. The equation to use is

$$\text{DATA SET SIZE IN BYTES} = (28 + \text{LONGROW}) * \text{NUMROWS}$$

NUMROWS is the number of rows to be contained in the data set and LONGROW is the length of the longest in the table space. For DSNDB06 table spaces, the value for LONGROW can be determined by running the following SQL statement:

```
SELECT MAX(RECLENGTH)
FROM   SYSIBM.SYSTABLES
WHERE  DBNAME = 'DSNDB06'
AND    TSNAME = 'name of table space to REORG'
AND    CREATOR = 'SYSIBM';
```

2. Ensure incompatible operations are not executing.
3. Start database DSNDB01 and DSNDB06 for read only access.
4. Run QUIESCE and DSN1CHKR utilities.
5. Take a full image copy of entire DB2 catalog and directory table spaces.
6. Start DSNDB01 and DSNDB06 for utility access.
7. Execute REORG utility.
8. Take a full image copy of entire DB2 catalog and directory table spaces.
9. Start table space and associated indexes for read/write access.

Furthermore, keep in mind that these six table spaces cannot be reorganized using specifying SHRLEVEL CHANGE. And finally, the SORTDATA, SORTDEVT, SORTNUM, and SORTKEYS options are ignored for these table spaces.

Steps to REORG Regular Table Spaces

The following steps should be used when reorganizing the remaining “regular” system catalog and directory table spaces:

1. Calculate the size of the unload data set (SYSREC) using the normal calculation:

$$\text{DATA SET SIZE IN BYTES} = \text{LONGROW} * \text{NUMROWS}$$

In this case it is unnecessary to add the additional 28 bytes to the length of the longest row. This is because these system catalog table spaces do not utilize links.

2. Ensure that incompatible operations are not concurrently executing (see the next section for an explanation of incompatible operations).
3. Start the table space and its associated indexes for read-only access.
4. Run CHECK INDEX on all indexes associated with the table space that is being reorganized.

5. Take a full image copy of the entire DB2 catalog and directory table spaces.
6. Start the table space and its associated indexes for utility access.
7. Execute the REORG utility.
8. Take a full image copy of the entire DB2 catalog and directory table spaces.
9. Start the table space and any associated indexes for read/write access.

These steps should be familiar to you because they closely follow the steps executed during the reorganization of an application data table space. There are several additional required steps added as precautions because of the critical nature of the DB2 catalog and directory.

NOTE

It is important to take a full image copy before and after reorganizing any DB2 catalog or directory table space.

Catalog Reorganization Restrictions

In addition to the procedures outlined previously, there are several restrictions on the manner in which the REORG TABLESPACE utility can be used with system catalog table spaces. First, recall that the SYSUTILX and SYSLGRNX table spaces in the DB2 Directory cannot be reorganized.

When reorganizing the DB2 Catalog (DSNDB06) and DB2 Directory (DSNDB01) table spaces, the following options cannot be used:

- The UNLOAD ONLY option is not permitted.
- Online REORG is not permitted for catalog and directory table spaces with links.
- The LOG YES option is not permitted as image copies are explicitly required following a catalog and/or directory reorganization.

Also, the reorganization of two specific table spaces are treated differently than any other in the manner in which they are tracked by DB2. Generally, DB2 records the reorganization of any table space in the SYSIBM.SYSCOPY system catalog table. However, DB2 records the reorganization of the DSNDB06.SYSCOPY and DSNDB01.DBD01 table spaces in the log instead.

You cannot collect inline statistics on the following DB2 Catalog and DB2 Directory table spaces:

DSNDB06.SYSDBASE	DSNDB06.SYSDBAUT
DSNDB06.SYSGROUP	DSNDB06.SYSPLAN
DSNDB06.SYSVIEWS	DSBDB06.SYSSTATS
DSNDB06.SYSHIST	DSNDB01.DBD01

Finally, in many 24x7 environments, it may be necessary to reorganize the system catalog and dictionary while it is being accessed. However, because of the central nature of the system catalog and directory to the operation of DB2, the following restrictions apply to concurrent activity during catalog reorganization:

- ALTER, DROP, and CREATE statements cannot be executed during the reorganization of any DB2 catalog or DB2 directory table space with the exception of SYSIBM.SYSSTR and SYSIBM.SYSCOPY.
- The BIND and FREE commands cannot be issued when the following table spaces are being reorganized: SYSIBM.SYSDBAUT, SYSIBM.SYSDBASE, SYSIBM.SYSGPAUT, SYSIBM.SYSPKAGE, SYSIBM.SYSPLAN, SYSIBM.SYSSTATS, SYSIBM.SYSUSER, and SYSIBM.SYSVIEWS.
- No DB2 utility can be running while SYSIBM.SYSCOPY, SYSIBM.SYSDBASE, SYSIBM.SYSDBAUT, SYSIBM.SYSSTATS, and/or SYSIBM.SYSUSER are being reorganized.
- No plan or package may be executed during the reorganization of SYSIBM.SYSPLAN and SYSIBM.SYSPKAGE.
- The GRANT and REVOKE statements cannot be issued when REORG is being run on SYSIBM.SYSDBASE, SYSIBM.SYSDBAUT, SYSIBM.SYSGPAUT, SYSIBM.SYSPKAGE, SYSIBM.SYSPLAN, and/or SYSIBM.SYSUSER.

The ability to reorganize the DB2 catalog and directory table spaces provides the DBA with a potent tool for his or her system tuning arsenal.

Take the Proper Image Copies

Finally, be sure to take a full image copy both before and after reorganizing any DB2 Catalog or DB2 Directory object. Failure to do so can result in an unrecoverable system table—a situation which is best avoided at all costs.