WEB 1

Special Consideration for DB2 Version 6

IN THIS CHAPTER

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

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.

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 = ' ';
```

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 procedures 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.