APPENDIX G

DB2 Version 8 Overview

IN THIS APPENDIX

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

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

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 `SAVEPOINT`s 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.