Craig S. Mullins

Return to Home Page

June/July 2003

The Resource for Users of IBM zSeries & S/390 Systems

z/JOURNAL

zDataPerspectives

by Craig S. Mullins

DB2 and the Old Dipsy Doo

By Craig S. Mullins

One of the biggest problems DBAs face when they are managing large partitioned DB2 table spaces is contending with non-partitioned indexes. Well, here comes IBM on its white horse with DB2 Version 8 to fight those problems with the old dipsy doo; in this case, dipsy is more appropriately spelled "DPSI."

However, before we examine the solution, let's first investigate the problem in a little more detail.

Non-Partitioned Indexes (NPIs)

In order to define a partitioned table space in DB2, a partitioning index is required. The CREATE INDEX statement specifies the range of values that DB2 will store in each specific partition. The partitioning index will have individual PART clauses, each which specifies the highest value that can be stored in the partition. To illustrate, consider Figure 1, which shows the CREATE statement for a partitioning index. This creates four partitions. Behind the scenes, DB2 will create four separate data sets – both for the table space data and for the index data. However, every other index defined on the table will be a "regular" DB2 index – that is, a non-partitioning index (NPI). This index resides in a single data set unless the PIECESIZE clause is used to break it apart – and even then the data will not be broken apart by partition.

Figure 1 – The CREATE Statement for a Partitioning Index

```
CREATE INDEX XEMP2
ON DSN8710.EMP (EMPNO ASC)
USING STOGROUP DSN8G710
PRIQTY 36 ERASE NO CLUSTER
 (PART 1 VALUES('H99'),
  PART 2 VALUES('P99'),
  PART 3 VALUES('Z99'),
  PART 4 VALUES('999'))
BUFFERPOOL BP1
CLOSE YES
COPY YES;
```

NPIs can cause contention, particularly with DB2 utilities. You can run a utility against a single table space or index partition, but you do not have that luxury with NPIs because they are not partitioned. You can minimize and manage downtime by running utilities a partition at a time. However, running utilities against NPIs can impact an entire table space. Additionally, contention on NPIs can cause performance bottlenecks during parallel update, insert, and delete operations.

Data Partitioned Secondary Indexes (DPSIs)

In Version 8 of DB2, IBM introduces the Data Partitioned Secondary Index (usually shortened to DPSI and pronounced "dipsy"). DPSIs are significant because they help to resolve the problems involved with NPIs that I just discussed. A DPSI is basically a partitioned NPI.

Therefore, 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 – even though the DPSI is created based on columns that are different from those used to define the partitioning scheme for the table space. Therefore, partition 1 of the DPSI will be for the same rows as partition 1 of the table space, and so on.

These changes to DB2 V8 provide many benefits including:

- The ability to cluster by a secondary index
- The ability to drop and rotate partitions easily
- Potentially less overhead in data sharing.

NPIs historically have caused DB2 performance and availability problems, especially with utilities. DPSIs solve many of these problems. With DPSIs there is an independent index tree structure for every partition. This means that utilities do not have to share pages or index structures. In addition, logical drains on indexes are now physical at the partition level. This helps utility processing in several useful ways. For example, you can run a LOAD by partition with no contention because the DPSI is partitioned the same way as the data and the partitioning index. Additionally, when reorganizing with DPSIs, the BUILD2 phase is not needed. Even your recovery procedures may be aided because you can copy and recover a single partition of a DPSI.

However, DPSIs are not magical objects that solve all problems. Indeed, changing an NPI to a DPSI may cause some queries to perform worse than before. Some queries will need to examine multiple partitions of the DPSI as opposed to the single NPI it previously used. On the other hand, if the query has predicates that reference columns in a single partition only, then performance may improve because only one DPSI partition needs to be probed.

Of course, not every index on a partitioned table should be a DPSI. You need to analyze your data access and utility processing requirements to determine when to use NPIs vs. when to use DPSIs. Before using DPSIs, you will have to examine your queries to determine predicate usage and the potential performance impact.

Summary

This introduction to DPSIs is not comprehensive. Be sure to investigate DPSIs over the next year so you will be ready to use them when DB2 V8 becomes generally available. By giving us the old dipsy-doo in Version 8, IBM is solving one of the bigger availability issues associated with DB2.

From zJournal, June/July 2003.