61

November 1997

In this issue

DB2 update

DB2 Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: xephon@compuserve.com

North American office

Xephon
1301 West Highway 407, Suite 201-450
Lewisville, TX 75067, USA
Telephone: 940 455 7050

Australian office

Xephon/RSM
PO Box 6258, Halifax Street
Adelaide, SA 5000
Australia
Telephone: 08 223 1391

Contributions

If you have anything original to say about DB2, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you be actively helping the free exchange of information, which benefits all DB2 users, but you will also gain professional recognition for your expertise, and the expertise of your colleagues, as well as some material reward in the form of a publication fee – we pay at the rate of £170 ($250) per 1000 words for all original material published in *DB2 Update*. If you would like to know a bit more before starting on an article, write to us at one of the above addresses, and we'll send you full details, without any obligation on your part.

Editor

Trevor Eddolls

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £235.00 in the UK; $350.00 in the USA and Canada; £241.00 in Europe; £247.00 in Australasia and Japan; and £245.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £20.00 ($30.00) each including postage.

DB2 Update on-line

Code from *DB2 Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Printed in England.

Large partitioned tablespaces

Tablespace partitioning, though new to some RDBMS products (such as Oracle 8), has been a feature of DB2 since its inception. Using a DB2 partitioned tablespace, data can be divided into components called 'partitions'. Because each partition resides in a separate physical dataset, DBAs can easily optimize the physical placement of data.

PARTITIONED TABLESPACE REVIEW

There are three types of DB2 tablespace, each one useful in different circumstances. The three types are:

- Simple tablespaces
- Segmented tablespaces
- Partitioned tablespaces.

Partitioned tablespaces are designed to increase the availability of data in large tables. For example: start and stop commands can operate at the partition level; partitioning enables and optimizes parallel query operations; partitions can be skipped during a tablespace scan; DB2 utilities can operate independently on separate partitions; DDL parameters such as free space, quantity, and bufferpool can be specified differently for each partition; and each partition can be placed on a different volume to optimize access by decreasing head contention.

However, partitioning does not come without a cost. First and foremost, a partitioned tablespace can contain only one table whereas segmented and simple tablespaces can contain multiple tables. Other potential drawbacks include the length of the partitioning key (which is limited to no more than 40 bytes), and the inability to update the columns of the partitioning index directly. To change a value in one of these columns, you must delete the row and then reinsert it with the new values. Finally, once defined, the partitioning key for the tablespace is difficult to change. Individual partitions cannot be removed or modified without dropping the partitioning index and starting over.

Prior to DB2 V5, the number of partitions restricted the maximum size of the dataset partition as shown in Figure 1.

Number of partitions	Maximum dataset size
1 to 16	4 GB
17 to 32	2 GB
33 to 64	1 GB

Figure 1: Partitions and dataset sizes

These limits still hold for non-large tablespaces even after DB2 V5. However, as we will soon learn, the limits differ for large tablespaces.

Deciding to use a partitioned tablespace is not as simple as merely determining the size of the table. Application-level factors (such as data contention, performance requirements, and the volume of updates to columns in the partitioning index) must be taken into account in the decision to use partitioned tablespaces. In general, use partitioned tablespaces in the following circumstances:

- When you wish to encourage parallelism.
- When the amount of data to be stored is very large (more than 1 million pages).
- To reduce utility processing time and decrease contention.
- To isolate specific data areas in dedicated datasets.
- To improve data availability. For example, if the data is partitioned by region, the partitions for the Eastern, Southern, and Northern regions can be made available while the Western region partition is being reorganized.
- To improve recoverability. Once again, if the data is partitioned by region and an error impacts data for the Eastern region only, then only the Eastern partition needs to be recovered.

DB2 VERSION 5 AND LARGE TABLESPACES

DB2 Version 5 provides a new partitioned tablespace feature known as 'large' tablespaces. A large tablespace can be created by using, appropriately enough, the LARGE parameter in the CREATE or ALTER TABLESPACE statement.

The LARGE parameter, available for partitioned tablespaces only, enables approximately 1TB of data to be stored in a single tablespace (albeit one consisting of multiple underlying VSAM datasets). The maximum size of a DB2 tablespace prior to DB2 Version 5 was 64GB.

Even though the marketing literature for DB2 Version 5 touts the ability to store up to a terabyte of data in a single tablespace, this limit will probably not be reached by most DB2 shops in the immediate future. Though it is true that multi-terabyte data warehouses and applications are being developed, this data usually spans multiple tables and typically includes the space utilized by accompanying indexes.

Instead, the primary benefit of large partitioned tablespaces will be additional flexibility. This is typified by the ability of large tablespaces to specify up to 254 partitions, each containing up to 4GB of data. When LARGE is not specified, the maximum number of partitions is the same as for previous DB2 releases – 64 partitions at 1GB each. As shown in Figure 1, the limitation on the size of each partition varies from 4GB to 1GB as the number of partitions increases.

Technical tip: if the NUMPARTS parameter is defined to be greater than 64, the tablespace will automatically be defined as a large tablespace even if the LARGE parameter is omitted.

USING LARGE TABLESPACES

Now that we know what a large tablespace is, let's examine the factors that allow us to decide when to use a large partitioned tablespace instead of a regular partitioned tablespace.

Space considerations and resource requirements need to be taken into account before creating a large tablespace. RIDs in a large tablespace are 5 bytes instead of 4 bytes. This causes an increase in the storage

required, impacting indexes in particular. A secondary consideration is the overall number of datasets required. Because large tablespaces can use more partitions, and each partition requires another dataset, overall resource consumption can increase.

Given these constraints, do not use the LARGE parameter without a valid reason. The two best motivating factors should be when more than 64 partitions are required or when more than 64GBs of data must be stored in a single tablespace.

Technical tip: you cannot alter existing partitioned tablespaces to be large partitioned tablespaces. Instead, you must unload the data, save existing granted authorizations, drop the tablespace, re-create the tablespace using the LARGE parameter, reload the data, and rebind any impacted packages and plans. This can be difficult to do without tools from third-party vendors.

IMPLEMENTATION IDEAS

The increased number of partitions available when using large tablespaces provides flexibility for database designers. Consider, for example, the design decisions that need to be made when a tablespace needs to store more than five years of data. The initial urge is to use the date as a partitioning key and specify one partition per month. However, with a limit of 64 partitions, this design technique limits the data to no more than 64 months (or 5 years and 4 months). Because DB2 V5 raises the limit to 254 partitions, the database design need not be tampered with. (Typical design ploys include forcing multiple months into a single partition, using multiple tables, or archiving data before it is practical to do so.)

OVERALL PROS AND CONS OF PARTITIONING

Advantages of a partitioned tablespace:

- Each partition can be placed on a different DASD volume to increase access efficiency.
- Partitioned tablespaces are the only type of tablespace that can hold more than 64GB of data (the maximum size of simple and

segmented tablespaces). As of DB2 V5, a partitioned tablespace can hold up to 1TB of data.

- Start and stop commands can be issued at the partition level. By stopping only specific partitions, the remaining partitions are available to be accessed, thereby promoting higher availability.
- Free space (PCTFREE and FREEPAGE) can be specified at the partition level enabling the DBA to isolate data 'hot spots' to a specific partition and tune accordingly.
- Query I/O, CPU, and Sysplex parallelism enable multiple engines to access different partitions in parallel, usually resulting in reduced elapsed time.
- Tablespace scans on partitioned tablespaces can skip partitions that are excluded, based on the query predicates.
- The clustering index used for partitioning can be set up to decrease data contention. For example, if the tablespace will be partitioned by DEPT, each department (or range of compatible departments) could be placed in separate partitions. Each department is in a discrete physical dataset, thereby reducing inter-departmental contention due to multiple departments co-existing on the same data page. Note that contention remains for data in non-partitioned indexes (although this contention has been significantly reduced by DB2 Version 4 and Version 5).
- DB2 creates a separate compression dictionary for each tablespace partition. Multiple dictionaries tend to cause better overall compression ratios. In addition, it is more likely that the partition-level compression dictionaries can be rebuilt more frequently than non-partitioned dictionaries. Frequent rebuilding of the compression dictionary can lead to a better overall compression ratio.
- The REORG, COPY, and RECOVER utilities can execute on tablespaces at the partition level. If these utilities are set to execute on partitions instead of on the entire tablespace, valuable time can be saved by processing only the partitions that need to be reorganized, copied, or recovered. Partition independence and

resource serialization further increase the availability of partitions during utility processing.

Disadvantages of a partitioned tablespace:

- Only one table can be defined in a partitioned tablespace.
- The entire length of the key for the partitioning index cannot exceed 40 bytes.
- The columns of the partitioning index cannot be updated. To change a value in one of these columns, you must delete the row and then re-insert it with the new values.
- The range of key values for which data will be inserted into the table must be known and stable before you create the partitioning index. To define a partition, a range of values must be hard-coded into the partitioning index definition. These ranges should distribute the data (more or less) evenly throughout the partitions. If you provide a stop-gap partition to catch all the values lower (or higher) than the defined range, monitor that partition to ensure that it does not grow dramatically or cause performance problems if it is smaller or larger than most other partitions.
- After you define the method of partitioning, you cannot change it easily. Individual partitions cannot be deleted or redefined. To drop the index that defines the partitioning, you must drop the table to which the index applies.

SUMMARY

Large partitioned tablespaces offer many benefits to DB2 users particularly those with immediate VLDB and data warehousing needs. However, the flexibility offered by large partitioned tablespaces will prove quite useful for DB2 applications that do not have huge storage requirements, but can benefit from additional partitions.

Craig S Mullins
VP Operations
Platinum Technology (USA)