



**CCAPRINT**

**A Newsletter Excerpt for Model 204 Users**

*February 2006*

USE OF AND ACCESS TO PRODUCTS AND FEATURES ARE IN ACCORDANCE WITH THE TERMS AND CONDITIONS OF THE USER'S SOFTWARE LICENSE. THE PRESENTATION OF MATERIAL HEREIN DOES NOT, IN ANY MANNER, MODIFY SUCH TERMS AND CONDITIONS.

## ***Dynamically Rebuilding the Reuse Queue with the BLDREUSE Command***

*By James Damon*

As part of our ongoing commitment to increase system availability, CCA implemented the BLDREUSE command in V6R1.0. This command provides for the dynamic, online rebuilding of the reuse queue in a Model 204 file that was defined as unordered, reuse record numbers (RRN). You can run the command even though the file is opened for update by many users. You do not have to stop application subsystems or bump users who have the file opened. However, before I describe the command, I need to explain the reuse queue.

### **The Reuse Queue**

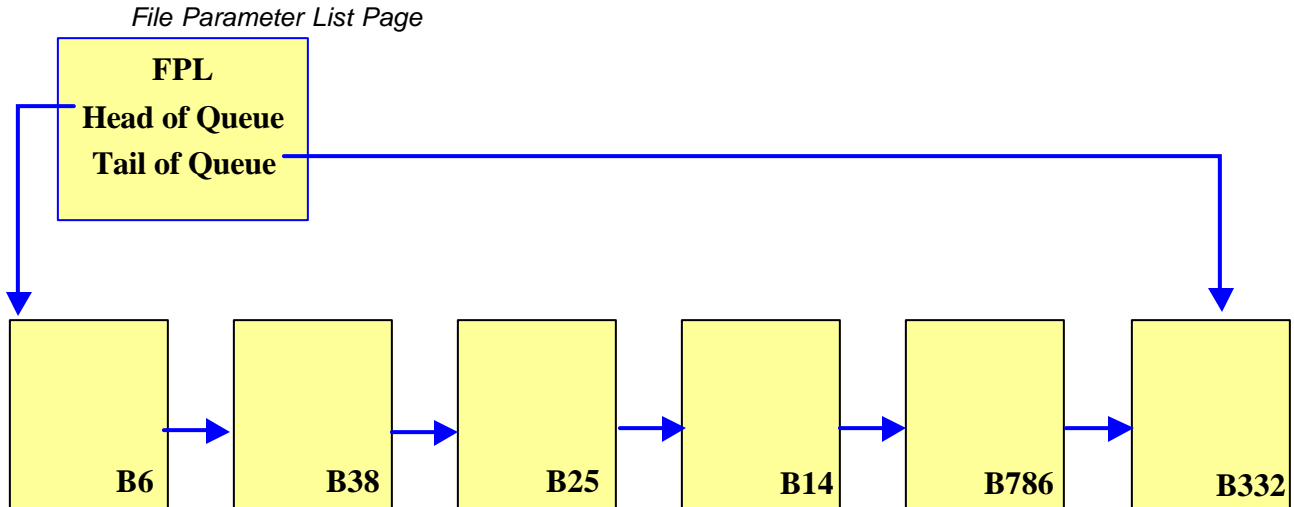
When a record is deleted with the DELETE RECORD statement in a FOR EACH RECORD loop, the Table B space consumed by the record is reclaimed and all related index entries are updated. This is not the case for the DELETE RECORDS IN statement (a logical delete) but that's a different topic. Reclaimed Table B space is always reusable. However, the record number made available by a DELETE RECORD statement is not reusable unless the file has been created as an RRN file. This is achieved at file CREATE by setting the file parameter FILEORG=X'24'.

An RRN file is unordered because new records may use the record numbers of previously deleted records. Therefore, the order of records in an unordered file is unpredictable; the order is neither chronological nor entry order, but instead unordered. The advantage of this file type is that in a volatile file with many record deletions, the storing of new records does not necessarily increase the physical size of Table B. The RRN feature helps to minimize the number of file segments, increases Table B utilization and at the same time reduces the risk of filling Table B.

## Introducing the Reuse Queue

The RRN feature is implemented with the construction of a queue of Table B pages known as the reuse queue. These pages are linked together by a pointer within each page pointing to the next page in the chain. Each page, when initially placed on the queue, must have at least one free record number and contain the minimum amount of free space (expressed as a percentage of 6144 bytes) required by the BREUSE file parameter. Figure 1 shows the schematic of the reuse queue.

Figure 1. Reuse Queue schematic



You can display the number of pages in this reuse queue by issuing the VIEW BQLEN command, as shown in Figure 2.

Figure 2. VIEW BQLEN output

```
VIEW BQLEN
BQLEN      6      TABLE B QUEUE LENGTH
```

## Reuse Queue in Operation

### DELETE RECORD Processing

At the completion of a DELETE RECORD statement, if the Table B page where the record used to reside contains at least BREUSE amount of free space, then that page is added to the tail of the reuse queue through the following sequence of events:

1. The next-page pointer on the previous Tail of Queue page is updated to point to the new tail page
2. The next-page pointer on the new tail page is set to minus one
3. The Tail of Queue pointer in the FPL is updated to point to the new tail page.

## STORE RECORD Processing

During a STORE RECORD operation against an RRN file, BHIGHPG is the first page used to store new records.

- If that page is full, the reuse queue is scanned starting with the Table B page at the head of the queue. Up to five pages on the queue are scanned looking for a page that can accommodate the new record.
- If no pages are found, BHIGHPG is incremented by one and the record is stored there.
- If BHIGHPG cannot be incremented by one because Table B is full, up to 200 pages will be selected at random from the reuse queue in an attempt to store the record. If the record still cannot be stored, the following message is issued and the file is marked full.

```
M204.1230: TABLE B FULL -- APPENDS --: filename
```

## Maintaining the Reuse Queue

Over time, a page on the reuse queue may no longer be eligible to be on the queue, because of changes in free space on the page. This can happen as records expand and consume more space on the page. However, pages are removed from the reuse queue only when a STORE RECORD statement attempts to use the page and fails due to insufficient space.

Other pages that are eligible may not be on the queue for the same reason. A record was deleted from a page but at the time the page had insufficient free space to be eligible for the queue. But, over time, free space has become available due to field deletion or field contraction. Pages are added to the reuse queue only during DELETE RECORD statements so many eligible pages in this state may not be known.

Changes to file parameters such as BRESERVE or BREUSE may also change the eligibility requirements for adding pages to or removing pages from the reuse queue. Prior to V6R1.0, file reorganization was the only way to rebuild the reuse queue.

## Introducing the BLDREUSE Command

The BLDREUSE command is available in V6R1.0 for dynamically rebuilding the reuse queue during normal system operation. You no longer have to stop subsystems or bump users. In other words, system availability is not affected by rebuilding the reuse queue in files where it is required. The syntax of the BLDREUSE command is illustrated in Figure 3.

Figure 3. BLDREUSE command syntax

```
BLDREUSE {NEW | [FROM nn][TO nn]}  
  
or  
  
IN filename BLDREUSE {NEW | [FROM nn][TO nn]}
```

The NEW keyword indicates that the reuse queue is to be completely rebuilt by scanning every Table B page to determine its eligibility for the queue. Use of the NEW keyword requires exclusive access to the file and can only be used when no other users have the file open.

The FROM and TO keywords do not require exclusive file access. If a BLDREUSE command is issued without keywords, a FROM 0 TO BHIGHPG clause is the default. Otherwise, only Table B pages in the range specified are examined for eligibility. Pages are added to the queue only when the FROM and TO keywords are used. (However, ineligible pages are not removed.)

Figure 4 illustrates two results of the BLDREUSE command after resetting the BREUSE parameter.

Figure 4. BLDREUSE output examples

```
VIEW BREUSE
BREUSE      20          FREE SPACE REQUIRED TO REUSE TABLE B PAGE
RESET BREUSE 10
BREUSE      10          FREE SPACE REQUIRED TO REUSE TABLE B PAGE

BLDREUSE NEW
TABLE B QUEUE LENGTH BEFORE REBUILD: 33
NUMBER OF PAGES THAT WERE ON QUEUE: 33
TABLE B QUEUE LENGTH AFTER REBUILD: 7857

=====

RESET BREUSE 5
BREUSE      5          FREE SPACE REQUIRED TO REUSE TABLE B PAGE
BLDREUSE NEW
TABLE B QUEUE LENGTH BEFORE REBUILD: 7857
NUMBER OF PAGES THAT WERE ON QUEUE: 7857
TABLE B QUEUE LENGTH AFTER REBUILD: 25053
```

### In Summary

If you think a file could be approaching a Table B full condition and many records are being stored and deleted, first view BQLEN. If the number is small or you think that there are more pages which should be eligible for the reuse queue, issue the BLDREUSE command. You could experiment by first using the FROM and TO keywords against, say one tenth or one quarter of the file. If the results indicate that significant numbers of pages were added to the queue, it may be worthwhile to use the NEW keyword and rebuild the entire queue. You may also want to reduce the value of the BREUSE file parameter to ensure that more pages are eligible for placement on the queue. Note that the command places an additional burden on the disk buffer pool, so running it during off hours is probably a prudent approach.