



CCAPRINT

A Newsletter Excerpt for Model 204 Users

January 2007

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.

COMPACTB – Compacting Table B by Combining Extension Records

By James Damon

Over time, files that are heavily updated tend to accumulate extension records or even chains of extension records where one base record has multiple extensions. As more extension records are created, performance slows and Table B becomes poorly utilized. Furthermore, extension records consume record numbers, which reduces the total number of logical records--base record, plus extensions--that can be stored in the file. This reduces the number of logical records that are possible for you to store. (The maximum number is 16,777,216 logical records in one file.)

Prior to V6R1.0, reorganizing a file is the only way to combine extension records with base records, thus improving performance and reclaiming record numbers for use as logical records. However, reorganization requires removing the file from service during the reorganization process, so this approach is generally suitable only for overnight batch processing. This, of course, assumes that there are times when the file can actually be removed from service. If you are attempting to implement a 24/7 environment, file reorganizations may not be feasible until the Online is brought down for other reasons.

Introducing the COMPACTB Command

V6R1.0 introduced the COMPACTB command, which provides you with a way to improve file performance and reduce extension records without taking the file out of service, as you would to perform a complete file reorganization. The COMPACTB command is restricted to files with the unordered file organization, where FILEORG=X'20'. This includes Reuse Record Number (RRN) files, where FILEORG=X'24', but excludes entry-order, hashed, and sorted files.

Reviewing COMPACTB Syntax

The syntax for the COMPACTB command is shown in Figure 1.

Figure 1. COMPACTB command syntax

```
COMPACTB [FROM ssss] [TO eeee] [FREE nn] [MAXE nn]
```

The options and variables shown in Figure 1 do the following:

- *FROM* ssss indicates the first Table B page number where the compactor starts looking for extensions. The default is zero.
- *TO* eeee indicates the final Table B page number where the compactor stops working. The compactor will finish scanning at the last record on this page. The default is the current highest active page number (BHIGHPG).
- *FREE* nn indicates the percentage of unused Table B pages (BSIZE minus BHIGHPG) the compactor may use for new extension records. The default is ten percent.
- *MAXE* nn specifies the percentage of page size (6144 bytes) that defines the maximum extension record size eligible for compaction. Larger extensions are not compacted. The default is 80%.

COMPACTB Processing Considerations

As the compactor runs, it issues a COMMIT statement after each record is compacted. However, transaction backout logging is not performed so COMPACTB processing is not a backoutable update unit. In the event of a failure, at most one record compaction is lost. Data, however, is never lost.

Since compaction is a CPU and I/O intensive process, we recommend that you compact files during off peak hours. However, CCA recognizes that may not always be possible. Therefore, to minimize the effect of the compactor on other logged-in users and to prevent it from monopolizing Online resources, the compactor

1. Periodically checks whether the issuing user has been bumped.
2. Relinquishes the CPU every 30 records to let other, higher priority users run.

A preimage of each page containing a compacted record is logged to the CHKPOINT dataset just as any other updated page is logged. And, a journal record is written to CCAJRNL for each compacted record. You may need to increase the sizes of the CHKPOINT and CCAJRNL datasets.

Each record to be compacted is locked exclusively, one-record-at-a-time. The lock is released when the update is committed. If the record lock is unavailable the record is skipped and not compacted during this attempt.

COMPACTB Processing in Action

In Figure 2, we will look at a few file parameters, both before and after the COMPACTB command is issued, to explain the results. We know from a dump of each Table B page that seven base records each had two extensions. The remaining 13 base records each had one extension.

Figure 2. Examining the results that occurred by compacting extension records

```
>IN TEST1 VIEW BHIGHPG,EXTNADD,EXTNDEL,BQLEN
BHIGHPG 39 TABLE B HIGHEST ACTIVE PAGE
EXTNADD 42 EXTENSION RECORDS ADDED
EXTNDEL 15 EXTENSION RECORDS DELETED
BQLEN    2 TABLE B QUEUE LENGTH

>IN TEST1 COMPACTB FROM 0 TO 9999999 FREE 100 MAXE 100
NUMBER OF BASIC RECORDS PROCESSED: 20
NUMBER OF EXTENSION RECORDS BEFORE COMPACTION: 27
NUMBER OF EXTENSION RECORDS AFTER COMPACTION: 20
NUMBER OF NOT PROCESSED (LOCKED) RECORDS: 0
NUMBER OF FREE PAGES USED: 2

>IN TEST1 VIEW BHIGHPG,EXTNADD,EXTNDEL,BQLEN
BHIGHPG 41 TABLE B HIGHEST ACTIVE PAGE
EXTNADD 49 EXTENSION RECORDS ADDED
EXTNDEL 29 EXTENSION RECORDS DELETED
BQLEN    5 TABLE B QUEUE LENGTH
```

The output from the first VIEW command shows that there were then 27 extension records (EXTNADD minus EXTNDEL). After the COMPACTB command completes we see that the number of extension records is now 20. From the second VIEW command we see that the reduction in extension records to 20 (EXTNADD minus EXTNDEL) resulted from adding seven extension records and deleting 14. That occurred because the two extensions on each of the seven base records were deleted (EXTNDEL plus 14) and then each of those two extensions were combined and restored as seven new extensions (EXTNADD plus seven).

In the process, we also consumed two free Table B pages to store the seven new extensions (BHIGHTPG plus two). And also, in the process of deleting the 14 extensions, we added three pages to the reuse queue (BQLEN plus three).

Combining Extension Records

When a base record has multiple extension records, the extension records are combined into fewer extension records. The result will never be less than one extension record.

The actual or existing size of extension records determines how the combination process works. (The maximum size of an extension record is 6144 bytes.) Consider the scenario illustrated in Figure 3. A single, base record has nine extension records with the lengths shown below. The lengths of the extensions in this example were chosen arbitrarily to illustrate how and when extensions are combined, or not combined.

Figure 3 is a schematic of a Table B logical record, with the base record and its extension records and record lengths, before compaction. Note that the Table B page numbers are not in numeric order because this is an RRN file.

Figure 3. Schematic of a Table B page logical record

B25	B470	B589	B642	B38	B320	B789	B84	B18	B953
Base record	Ext 1 40	Ext 2 1200	Ext 3 2400	Ext 4 3200	Ext 5 4300	Ext 6 2300	Ext 7 60	Ext 8 90	Ext 9 120

Figure 4 is the schematic of the same Table B logical record after compaction. The chain of extension records is reduced from nine to four.

1. Extensions 1, 2, and 3 have a total combined length of 3640 bytes, so they compact into one.
2. Extension 4 is left as is, because combining it with the previous page exceeds the page length limit.
3. Extension 5 is also left as is, because if combined with either Extension 4 or Extension 6, the resulting length would be greater than 6144 bytes.
4. Extensions 6,7,8, and 9 have a combined length of 2570 bytes, which compacts into one extension record.

Figure 4. The Figure 3 Table B logical record after compaction

B25	BHIGHPG	B38	B320	B348
Base record	Ext 1 3640	Ext 2 3200	Ext 3 4300	Ext 4 2570

In Conclusion

If you have files with a high number of extension records and would like to improve performance without reorganizing the file, consider using the COMPACTB command. You do not need to take the file out of service during the compaction process, although CCA recommends performing compaction during off-peak hours. If a significant number of extension records are deleted as a result, then file performance should improve and additional space in Table B will be made available for future growth.