

# SELECT<sup>®</sup> Online

The Journal of the International Oracle Users Group

Select Journal  
Home

Regular Columns

Past Issues

General Information

IOUG Home



## Select Magazine - October 2000

Volume 7, No. 1

### Rebuilding Large Indexes "In-Place"

By Wayne Linton, I.S.P.

*ORACLE7 V7.2 and later versions have introduced time saving functionality that can lighten the maintenance burden of the DBA. By utilizing the REBUILD, RESIZE and AUTOEXTEND features of the ORACLE database, the task of rebuilding large indexes can be automated. All it takes is to set up the table of information for each tablespace you wish to manage in this way, create the datafiles with the appropriate parameters and set up the job to run periodically.*

Every DBA is faced with having to regularly rebuild indexes on tables that incur a lot of DML activity. Over time, the indexes become fragmented and extend to the point where insufficient free space is left in the tablespace. Usually, a large index will need to be recreated in a new tablespace. The old tablespace is then dropped, but sometimes the old datafile remains allocated and cannot be deleted to free up the physical space on the disk. This can occur in 24-by-7 shops where a continuously running task is utilizing the index.

ORACLE7 V7.2 introduced the concept of REBUILD-ing indexes rather than recreating them, thus enabling the maintenance of in-use indexes and reducing table locking contention. The UNRECOVERABLE parameter improved performance, as did the PARALLEL parameter. Another useful feature, introduced in V7.3, is the ability to RESIZE and AUTOEXTEND individual datafiles. All of these features put together can form the basis for automating the rebuilding of large indexes "in-place."

#### "In-Place"?

Well, to be completely accurate about it, the rebuild does not occur in the same physical location as the original index. We can, however, manipulate two datafiles belonging to the same tablespace so that the rebuild occurs without having to go through the process of creating a new and dropping the old tablespace and datafile. So in this sense, the rebuild takes place in the same tablespace and the freed-up space can be released automatically.

#### The Technique

You can do this by placing large indexes into their own tablespace, consisting of one large datafile. A second very small datafile is created. Both are AUTOEXTENDable to a MAXSIZE of the same size, large enough to accommodate the index plus some reasonable amount of growth. Both have a NEXT set to the (MAXSIZE \_ 5M). The two datafiles need not be on the same volume.

A table of sizing parameters is created for each tablespace. LARGE\_SINGLE\_INDEX\_REBUILD stores the index's name, INITIAL, NEXT and parallel DEGREE parameters to be used in rebuilding it. Automated rebuilding will only occur for the tablespaces/indexes that are named in this table.

The script compares the largest extent of free space left in the tablespace with the NEXT extent of the index contained within. If there is not enough room for the next extent, the rebuild code is generated and run. Once the index has been rebuilt, the datafile just vacated is shrunk to 5m. So the rebuild isn't actually done 'in place', but it's pretty close! The important thing is that it is automated.

There is a chance that during the rebuild process, an extent of the new index may be allocated in the original datafile that would prevent shrinking it down to 5M at the end of the process. To prevent this, all remaining space over 5M in the tablespace is allocated to the index. When rebuilt in the new datafile, it will use the sizing information stored in the LARGE\_SINGLE\_INDEX\_REBUILD table.

## Stepping Through an Example

You must first identify those indexes that you wish to include in this rebuilding automation scheme. Typically, they will be in the gigabyte(s) range. [Note: be careful not to create datafiles larger than allowed \_ see ORACLE article <Note:112011.1>] Entries must then be made in the LARGE\_SINGLE\_INDEX\_REBUILD table for each tablespace. Only one index can be placed in each tablespace for this process to work as intended. Decide upon the degree of parallelism you wish to use in the rebuild. If you leave this column NULL, or set it to 0 or 1, parallelism will not be used.

In our example, we will create an index named SAMPLE\_INDEX of size 1200M in a tablespace named SAMPLE\_TABLESPACE of size 2000M. We will use parallel query of degree 3 to help us rebuild the index. The entry in the LARGE\_SINGLE\_INDEX\_REBUILD table would look something like this:

TABLESPACE_NAME	INDEX_NAME	STORAGE_INITIAL	STORAGE_NEXT	PARALLEL_DEGREE
SAMPLE_TABLESPACE	SAMPLE_INDEX	400M	50M	3

The space allocated for the index will be PARALLEL\_DEGREE extents of size STORAGE\_INITIAL. If this is not enough space, more extents of size STORAGE\_NEXT will be allocated. If less space is required, some portion of these extents will be returned to free space.

Create the tablespace to accommodate the index, plus some space for growth. Set the datafile as size 5m, auto-extendable to 2000M (so the next will be 1995M). The tablespace creation would look something like this:

```
CREATE TABLESPACE SAMPLE_TABLESPACE DATAFILE
'C:\ORACLE\ORADATA\dbname\SAMPLE01.DBF'
SIZE 5M AUTOEXTEND ON NEXT 1995M MAXSIZE 2000M;
```

Now create the index for the first time into this tablespace. For the purposes of this example, I will assume that it already exists somewhere else, so we will rebuild it into this new tablespace.

```
ALTER INDEX SAMPLE_INDEX REBUILD UNRECOVERABLE PARALLEL 3
STORAGE (INITIAL 400M NEXT 50M PCTINCREASE 0)
TABLESPACE SAMPLE_TABLESPACE;
```

Once completed, the tablespace will have expanded to 2000M. Now allocate a second datafile with the same characteristics as the first:

```
ALTER TABLESPACE SAMPLE_TABLESPACE ADD DATAFILE
'C:\ORACLE\ORADATA\dbname\SAMPLE02.DBF'
SIZE 5M AUTOEXTEND ON NEXT 1995M MAXSIZE 2000M;
```

The stage is now set. The SQL script in Figure 1 need only be scheduled to run on a regular basis. Indexes that cannot extend in the current datafile will be rebuilt into the new datafile using the storage parameters defined for it in the LARGE\_SINGLE\_INDEX\_REBUILD table. The space in the old datafile will be released.

```
REM -----
REM LARGE_SINGLE_INDEX_REBUILD.SQL
REM
REM This script will generate the SQL commands to rebuild an index
REM in place [sort of!]. The technique involves:
REM
REM 1) one tablespace with a datafile #1 containing one large index
REM 2) second datafile of size 5m extendable to the same size as
```

```

datafile #1
REM
REM i.e. ALTER DATABASE DATAFILE 'datafile2'
REM AUTOEXTEND ON NEXT 1995M MAXSIZE 2000M;
REM
REM 3) a table named LARGE_SINGLE_INDEX_REBUILD with index rebuild
REM sizing parameters as follows:
REM
REM CREATE TABLE LARGE_SINGLE_INDEX_REBUILD (
REM TABLESPACE_NAME VARCHAR2(30) NOT NULL,
REM INDEX_NAME VARCHAR2(30) NOT NULL,
REM STORAGE_INITIAL VARCHAR2(10) NOT NULL,
REM STORAGE_NEXT VARCHAR2(10) NOT NULL,
REM PARALLEL_DEGREE CHAR(1));
REM
REM ... here is some sample data:
REM
REM TABLESPACE INDEX INITIAL NEXT DEGREE
REM -----
REM SAMPLE_TBS1 SAMPLE_INDEX_1 500M 50M 0
REM SAMPLE_TBS2 SAMPLE_INDEX_2 400M 50M 3
REM SAMPLE_TBS3 SAMPLE_INDEX_3 250M 25M 4
REM
REM
REM 4) The script then generates the SQL to rebuild the index in the
REM small empty datafile, then resizes the original datafile to
REM 5m. The rebuild is triggered when the index's next extent
REM is larger than the free space left in the tablespace.
REM Before starting the rebuild into the new datafile, allocate
REM all free extents over 5m to force ALL of the rebuild into
REM the new datafile.
REM
REM Wayne Linton, OCP/DBA May, 2000
REM
REM -----

```

```

SET HEADING OFF ECHO OFF FEEDBACK OFF
SPOOL LSIR.SQL

```

```

SELECT 'ALTER INDEX '||I.OWNER||'.'||I.INDEX_NAME||
        'ALLOCATE EXTENT (SIZE '||F.BYTES||');'
FROM DBA_INDEXES I,
     DBA_FREE_SPACE F,
     LARGE_SINGLE_INDEX_REBUILD L
WHERE L.INDEX_NAME = I.INDEX_NAME
     AND L.TABLESPACE_NAME = I.TABLESPACE_NAME
     AND I.TABLESPACE_NAME = F.TABLESPACE_NAME
     AND F.BLOCKS > 2560 - USE UP ALL SPACE OVER 5 MEGS
     AND I.NEXT_EXTENT > (SELECT MAX(F.BYTES)
                          FROM DBA_FREE_SPACE F
                          WHERE F.TABLESPACE_NAME = L.TABLESPACE_NAME)
     AND 1 = (SELECT COUNT(INDEX_NAME)
              FROM DBA_INDEXES
              WHERE TABLESPACE_NAME = L.TABLESPACE_NAME)

```

```

/

```

```

SELECT 'ALTER INDEX '||I.OWNER||'.'||I.INDEX_NAME||
        ' REBUILD UNRECOVERABLE '||CHR(10)||
        DECODE(L.PARALLEL_DEGREE, 0,NULL, 1,NULL, NULL,NULL,
        ' PARALLEL '||L.PARALLEL_DEGREE)||
        ' STORAGE (INITIAL '||L.STORAGE_INITIAL||

```

```

        \ NEXT \||L.STORAGE_NEXT||' PCTINCREASE 0)'||CHR(10)||
        \ TABLESPACE \||L.TABLESPACE_NAME ||';'
    ||CHR(13)||CHR(10)||
    \ALTER DATABASE DATAFILE \||CHR(10)||
    \||D.FILE_NAME||\||CHR(10)||' RESIZE 5M;'
FROM DBA_INDEXES I,
     DBA_DATA_FILES D,
     LARGE_SINGLE_INDEX_REBUILD L
WHERE L.INDEX_NAME = I.INDEX_NAME
     AND L.TABLESPACE_NAME = I.TABLESPACE_NAME
     AND I.TABLESPACE_NAME = F.TABLESPACE_NAME
     AND D.BLOCKS > 2560 - FILE NOT SHRUNK TO 5 MEGS CONTAINING INDEX
     AND I.NEXT_EXTENT > (SELECT MAX(F.BYTES)
                          FROM DBA_FREE_SPACE F
                          WHERE F.TABLESPACE_NAME = L.TABLESPACE_NAME)
     AND 1 = (SELECT COUNT(INDEX_NAME)
              FROM DBA_INDEXES
              WHERE TABLESPACE_NAME = L.TABLESPACE_NAME)

/

SPOOL OFF
SET HEADING ON FEEDBACK ON ECHO ON
START LSIR

```

**Figure 1: LARGE\_SINGLE\_INDEX\_REBUILD.SQL**

Figure 2 is sample code generated from the LARGE\_SINGLE\_INDEX\_REBUILD.SQL script that, when run, will fill up the current datafile, rebuild the index in the new datafile and shrink the vacated datafile releasing the space.

```

ALTER INDEX owner.SAMPLE_INDEX ALLOCATE EXTENT (SIZE 23869184);
ALTER INDEX owner.SAMPLE_INDEX ALLOCATE EXTENT (SIZE 14788992);

ALTER INDEX owner.SAMPLE_INDEX REBUILD UNRECOVERABLE
     PARALLEL 3 STORAGE (INITIAL 500M NEXT 50M PCTINCREASE 0)
     TABLESPACE SAMPLE_INDEX;

ALTER DATABASE DATAFILE
'C:\ORACLE\ORADATA\dbname\SAMPLE02.DBF'
RESIZE 5M;

```

**Figure 2: Sample Generated SQL to Rebuild a Large Single Index**

## ORACLE8i Functionality

A new ALTER INDEX ... COALESCE option is introduced in ORACLE8i that can collect the fragmented space in the leaf blocks of a B-tree index. Although this gathers up free space for reuse at the leaf level and defers the index extending to get more space, it does not shrink the branches or depth of the B-tree. Only rebuilding or recreating the index will do this.

Partitioned indexes help with the performance and management of extremely large indexes by breaking them into smaller, more manageable pieces. Space management for each index partition behaves much like that of regular indexes. Each partition can be in its own tablespace and can be managed separately. They grow, extend, and will need to be rebuilt from time to time. If an index partition is created in its own tablespace, the technique outlined in this article can be extended to rebuild it "in-place." File name: Oracle Internet Commerce Server\_22.doc

## About the Author

**Wayne Linton** is an ORACLE Certified Professional DBA at Shell Canada Limited based in Calgary, Alberta. Wayne is the current president of the Calgary ORACLE User Group, and can be reached at [wayne.linton@shell.ca](mailto:wayne.linton@shell.ca).

[Download Acrobat Reader](#)

Copyright 2003 by the International Oracle Users Group