Reorganisation einer 10TB-Datenbank mit Einsatz der Advanced Compression Option

Susanne Jahr
Herrmann & Lenz Services GmbH, Burscheid

Schlüsselworte

Reorganisation, Komprimierung, Advanced Compression Option, Secure Files, LOBs, Redefinition

Einleitung

Für die produktive Auftragseingangs-Datenbank eines Kunden mit der Gesamtgröße von ca. 10 TB bestand Reorganisationsbedarf. Es gab zwei Projektziele:

1.: Verkleinerung der Datenbank und dadurch Freigabe nicht mehr benötigter ASM-Platten

2.: Implementierung eines Information Life Cycle mit verschiedenen Storage-Ebenen, um ältere, nur noch selten genutzte Daten weg vom kostenintensiven "High-Performance"-ASM-Plattenspeicher auf günstigere ASM-Plattenbereiche auszulagern

Ausgangssituation

Die Datenbank wird in einem 3-Knoten-RAC betrieben. Als Shared Storage wird ASM verwendet. Die Lizenzierung ist Enterprise Edition mit den Optionen RAC, Partitioning, Real Application Testing und Advanced Compression.

LOBs machen den Großteil des Gesamt-Datenvolumens in der Datenbank aus; die größten 10 Segmente belegten vor Projektbeginn ca. 3,6 TB in der Datenbank.

```
TABLESPACE_NAME    FILE_MB
-------------- ----------
XXX_CAD           7.170.000   → LOB-Tablespace
XXX_DAT             598.720
XXX_IDX             581.632
XXX_PDAT            565.248
XXX_PIDX          1.207.000
-------------- ----------
Summe:           10.122.600
```

Die vorhandenen LOBs liegen als Basic File LOBs vor. Um die Mechanismen der ACO anwenden zu können, müssen alle LOBs in SecureFile LOBs umgewandelt werden.

Die Datenbank hat eine Blockgröße von 16k. Der Tablespace für die LOBs hat jedoch eine 8k-Blockgröße.

Konzept

Um die Projektziele zu erreichen, wurden folgende Maßnahmen festgelegt:

1. Information Life Cycle Management: Aufteilung der Daten in
    a. Aktuell
    b. Älter
    c. Archiv
2. Definition und Konfiguration einer neuen ASM-Diskgruppe mit Platten aus dem günstigeren Speicherbereich. Somit Implementierung der zwei Storage-Ebenen "High-Performance" und "Archive" sowie Zuweisung der Daten aus den Bereichen "Älter" und "Archiv" zur Storage-Ebene "Archive"
3. Komprimierung der LOB- und Datensegmente unter Einsatz der Advanced Compression Option (ACO)
4. Gleichzeitige Reorganisation in die neuen Storage-Ebenen gem. Punkt 2
5. Teilweise "In-Place"-Reorganisation, um zusammenhängenden Free-Space zu gewinnen und damit nicht mehr benötigten Platten freigeben zu können
6. Sonderfall: Löschung von nicht mehr benötigten "Altlasten" aus einer Sammel-Tabelle

Durchführung

1. *Storage-Ebenen*
   Die Implementierung der neuen Storage-Ebene "Archiv" erfolgte durch Anlage eines neuen Tablespaces in einer eigenen ASM-Diskgruppe. Abweichend von den Default-Werten wurde in der Default-Storage Klausel ein Initial Extent von 8M definiert.

```
CREATE TABLESPACE XXX_ARCHDATA DATAFILE
   SIZE 10G AUTOEXTEND ON NEXT 1G MAXSIZE 1400G
BLOCKSIZE 8K
DEFAULT STORAGE (INITIAL 8M)
```

2. *LOB-Komprimierung*
   Die größte Datenmenge stellen partitionierte Tabellen mit großen LOB-Segmenten.
   Mit Hilfe des Packages dbms_redefinition wurden diese Tabellen partitionsweise reorganisiert, wobei die älteren Partitionen in den Archive-Tablespace verschoben werden sollten. Aufgrund der unterschiedlichen Blockgrößen der LOB- und der übrigen Spalten war dies jedoch nicht möglich, so dass lediglich die LOB-Segmente in den Archive-Tablespace verschoben wurden, womit jedoch der Hauptanteil des Platzbedarfs abgedeckt ist. Die übrigen Daten wurden In-Place reorganisiert. Für die LOB-Reorganisation wurde für jede Partition eine Zwischentabelle angelegt, wobei die LOB-Spalte(n) als SecureFile LOBs im neuen Archive-Tablespace definiert wurden. Der Ablauf war wie folgt:

    a. dbms_redefinition.can_redef_table → Ist die gewünschte Tabelle bzw. Partition bereit zur Redefinition?

```
EXEC dbms_redefinition.can_redef_table
(uname => 'PRODUKTIV', tname => 'T299_PRINTOUT', part_name => 'P0001')
```

b. Anlage der Zwischentabelle

```
CREATE TABLE "PRODUKTIV"."I299_PRINTOUT"
("T299_PRINTOUT_ID" NUMBER(10,0) NOT NULL ENABLE,
...
)
   TABLESPACE &TAB_TS STORAGE (INITIAL 8M)
   LOB ("T299_PLOT") STORE AS SECUREFILE "T299_PLOT_SFLOB"
      (TABLESPACE &LOB_TS STORAGE (INITIAL 8M) ENABLE STORAGE IN ROW
COMPRESS LOW RETENTION AUTO CACHE READS);
```

c. dbms_redefinition.start_redef_table → Start der Redefinition

```
EXEC dbms_redefinition.start_redef_table
(uname => 'PRODUKTIV', orig_table => 'T299_PRINTOUT', int_table =>
'I299_PRINTOUT', part_name => 'P0001')
```

d. dbms_redefinition.finish_redef_table → Beendigung der Redefinition, Tausch der Original- mit der Zwischentabelle

```
EXEC dbms_redefinition.finish_redef_table
(uname => 'PRODUKTIV', orig_table => 'T299_PRINTOUT', int_table =>
'I299_PRINTOUT', part_name => 'P0001')
```

3. *OLTP-Komprimierung*

Große Tabellen ohne LOB-Segmente wurden ebenfalls mit Hilfe der ACO komprimiert. Auch hier wurde eine Storage-Klausel von INITIAL 8M definiert. Die zugehörigen Index-Partitionen, die bei der Verschiebung der Tabellen-Partitionen UNUSABLE werden, wurden im Anschluss neu aufgebaut:

```
ALTER TABLE t038_option MOVE PARTITION P0001 COMPRESS FOR OLTP STORAGE
(INITIAL 8M NEXT 8M)
/

BEGIN
    FOR r IN (SELECT index_name, ip.partition_name FROM user_indexes i join
              user_ind_partitions ip using (index_name) WHERE i.table_name =
              'T038_OPTION' and ip.status = 'UNUSABLE')
    LOOP
        EXECUTE IMMEDIATE 'ALTER INDEX "' || r.index_name || '" REBUILD
PARTITION "' || r.partition_name || '" STORAGE (INITIAL 8M NEXT 8M)';
    END LOOP;
END;
/
```

4. *Spezialfall: "Altlasten"-Tabelle*

Eine sehr große, nicht partitionierte Tabelle enthält historische Daten über alle (auch bereits abgewickelte und fakturierte) Aufträge inklusive zugehörige Dokumente (u.a. pdf-Dateien; BLOB). Da bisher keine Bereinigung stattfand, weist diese Tabelle ein starkes Wachstum von 1 GB in 2007 auf ca. 856 GB zum Projektstart auf.

Die laut Informationen der Fachabteilung nicht mehr benötigten Datensätze können gelöscht werden. Hierbei kommt das Package `dbms_parallel_execute` zum Einsatz. Es werden anhand der Row-IDs Datenbereiche definiert, innerhalb derer mit Hilfe des Job Schedulers parallel das gewünschte Statement (in diesem Fall ein DELETE auf die Zieltabelle) ausgeführt wird.

```
DECLARE
del_t038 VARCHAR2(30) := 'del_t038';
l_sql_stmt VARCHAR2(32767);
l_try NUMBER;
l_status NUMBER;

BEGIN
-- Create the TASK
DBMS_PARALLEL_EXECUTE.create_task (task_name => del_t038);

-- Chunk the table by the ROWID
DBMS_PARALLEL_EXECUTE.create_chunks_by_rowid(task_name => del_t038,
          table_owner => 'PRODUKTIV',
          table_name => 'T038_OPTION',
          by_row => TRUE,
          chunk_size => 100000);

-- DML to be execute in parallel
l_sql_stmt := 'DELETE FROM produktiv.t038_option WHERE ...
             AND rowid BETWEEN :start_id AND :end_id';

-- Run the task
DBMS_PARALLEL_EXECUTE.run_task(task_name => del_t038,
sql_stmt => l_sql_stmt,
language_flag => DBMS_SQL.NATIVE,
parallel_level => 10);
```

Anschließend wurde das LOB-Segment der Tabelle In-Place migriert.

Fazit

Durch die verschiedenen Maßnahmen konnten bis jetzt bereits ca. 3,8 TB in der Datenbank eingespart werden. Ca. 2,6 TB LOB-Segmente wurden bisher in die Archive-Storage-Ebene verschoben.

Kontaktadresse

Susanne Jahr
Herrmann & Lenz Services GmbH
Höhestr. 37
51399 Burscheid

Telefon: +49 (0) 2174-6712-14
Fax: +49 (0) 2174-6712-22
E-Mail susanne.jahr@hl-services.de
Internet: www.hl-services.de