dbms_redefinition – damit kann man Vieles tun

Klaus Reimers
ORDIX AG
Paderborn

Schlüsselworte
Reorganisation – Redefinition – LOB – Compression – IOT – Partition - CTAS – DataPump – table move – dbms_redefinition – segment shrink

Einleitung

„Ich muss Überstunden am Wochenende machen, da ich diverse Tabellen in meiner Datenbank reorganisieren muss“. Diesen Satz habe ich in den letzten Jahren des öfteren gehört. Dem Administrator stellen sich somit folgende Fragen:
Ist es wirklich notwendig, so oft Tabellen zu reorganisieren, wie es in der Praxis gemacht wird?
Welche Vorgehensweise ist bei einer eventuellen Reorganisation am geeignetsten?
Benötigt man ein Wartungsfenster?
Welche der möglichen Varianten verwende ich?

Zunächst gebe ich einen Überblick über die gängigen Methoden und werde dann auf den eigentlichen Schwerpunkt meines Vortrags (dbms_redefinition) eingehen.

Methoden

CTAS

Hier erfolgt eine Reorganisation innerhalb der Datenbank.
Folgende Vorgehensweise muss gewählt werden:

- exklusiven Zugriff auf die zu reorganisierende Tabelle erzeugen
- Tabelle unter neuem Namen unter Veränderung der Storage-Parameter kopieren
- eventuell Foreign Keys ausschalten, die auf die Tabelle zeigen
- Originaltabelle löschen per TRUNCATE
- Daten per INSERT .. SELECT zurückschieben
- eventuell Foreign Keys, die auf die Tabelle zeigen, wieder einschalten

Die Reorganisation muss hier innerhalb eines Wartungsfensters durchgeführt werden.

DataPump

Folgende Vorgehensweise muss gewählt werden.

- exklusiven Zugriff auf die zu reorganisierenden Tabelle erzeugen
- Exportieren der zu reorganisierenden Tabelle mit
- Aushängen aller Foreign Key Constraints, die auf diese Tabelle zeigen
- Löschen der Tabelle per TRUNCATE
- Importieren des generierten Export Files
- Neuerstellung der Foreign Key Constraints

Die Reorganisation muss hier innerhalb eines Wartungsfensters durchgeführt werden.

Storage Klauseln sollten im alten Zustand modifiziert werden.

alter table move

Folgende Vorgehensweise muss gewählt werden:
Man erzeugt einen exklusiven Zugriff auf die zu reorganisierende Tabelle

- Die Reorganisation wird gestartet über

```
alter table <table_name>
     move tablespace <tablespace_name>
     storage (...);
```

- Rebuilden aller abhängigen Indizes über

```
alter index <index_name> rebuild;
```

Die einzelnen Zeilen werden in andere Extents verlagert, es wird also lediglich der Speicherort verändert. Alle abhängigen Objekte bleiben erhalten, lediglich die Indizes müssen ein „Rebuild“ erfahren.
Auch bei Verwendung dieser Funktionalität ist ein Wartungsfenster notwendig.
Mit Oracle 12.2 und Nutzung der Enterprise Edition kann man mit diesem Verfahren allerdings auch Online reorganisieren

```
alter table <table_name>
     move tablespace <tablespace_name>
     storage (...) online;
```

Segment Shrink

Seit Oracle 9i kann man Tablespaces mit der zusätzlichen Eigenschaft ASSM definieren (Automatic Segment Space Management). In Oracle 10g ist aufbauend auf diesem Feature eine neue Funktionalität hinzugekommen, die Möglichkeit Segmente zu verkleinern.
Um Segmente zu verkleinern, muss das ROW MOVEMENT eingeschaltet sein.

```
ALTER TABLE <table_name> ENABLE ROW MOVEMENT;
```

Über das Kommando

```
ALTER TABLE <table_name> SHRINK SPACE COMPACT;
```

werden die Sätze der Tabelle in die vorderen Blöcken verdichtet. Die High Water Mark bleibt allerdings am alten Platz.
Über das Kommando

```
ALTER TABLE <table_name> SHRINK SPACE CASCADE;
```

werden die Tabellen auch verdichtet, aber zusätzlich wird die High Water Mark nach vorne verschoben und eventuell freie Extents zurückgegeben.
Das Verschieben erfolgt intern über Insert/Delete. Die Indizes werden somit gepflegt und sind anschließend „USABLE“. Trigger werden nicht gefeuert.

dbms_redefinition

Nun komme ich zum Schwerpunkt meines Vortrages, den Möglichkeiten des Packages dbms_redefinition.

Diese Möglichkeit ist mit Oracle 9i hinzugekommen, sie wird auch als Online Table Reorg bezeichnet.

Normaler Ablauf
Folgende Vorgehensweise muss grundsätzlich gewählt werden:

- Erstellen einer neuen zunächst leeren Tabelle
- Starten der Reorganisation mittels

```
BEGIN
  dbms_redefinition.start_redef_table(
    '<schema>',
    '<table_orig>',
    '<table_int>')
END;
/
```

- Erzeugen aller notwendigen Objekte und Eigenschaften auf dieser Tabelle

```
DECLARE
  num_err    number;
BEGIN
  dbms_redefinition.copy_table_dependents(
    '<schema>',
    '<table_orig>',
    '<table_int>',
    num_errors => num_err);
END;
/
```

- Beenden der Reorganisation mittels

```
BEGIN
  dbms_redefinition.finish_redef_table(
    '<schema>',
    '<table_orig>',
    '<table_int>')
END;
/
```

- Löschen der Interimstabelle inklusive der abhängigen Objekte

Tabellen können somit online reorganisiert werden.
Wird die Tabellenstruktur verändert und sind davon abhängige Views betroffen, so müssen diese Views in ihrer Definition verändert werden.
Allerdings ist hier die Lizensierung der Enterprise Edition notwendig.

Ablauf feiner granuliert
Sind die Tabellen sehr groß oder soll individueller auf einzelne abhängige Elemente eingegangen werden, dann kann der Schritt 3 (Erzeugen abhängiger Objekte) feiner granuliert werden.

Anstelle der Prozedur dbms_redefinition.copy_table_dependents wird die Prozedur register_dependent_object verwendet.
Ich zeige das hier am Beispiel eines Index.

```
create unique index <index_int> on <table_int>
(<column_list>);
BEGIN
   DBMS_REDEFINITION.REGISTER_DEPENDENT_OBJECT
   ('<schema>', '<table_orig>', 'table_int',
   DBMS_REDEFINITION.CONS_INDEX,
   '<schema>', 'index_orig>', '<index_int>');
 END;
/
```

Für jedes Constraint (DBMS_REDEFINITION.CONS_CONSTRAINT) und jeden Trigger (DBMS_REDEFINITION.CONS_TRIGGER) müssen ebensolche Einträge gemacht werden. Berechtigungen (Grants) müssen lediglich gesetzt werden.

Veränderung von Spalten

Spalten können während der Reorganisation umbenannt, weggelassen, hinzugefügt oder im Datentyp verändern werden. Dies geschieht über den Parameter col_mapping in der Prozedur start_redef_table.

```
   BEGIN
   dbms_redefinition.start_redef_table
   (<schema>,'<table_orig>',<table_int>',
   col_mapping=>'<spalte1_orig spalte1_int, …');
   END;
   /
```

Wird ein Column Mapping durchgeführt, so muss jede einzelne Spalte aufgeführt werden, die in der Zieltabelle ankommen soll. Dabei sind auch Typkonvertierungen über Funktionen möglich oder aber auch das Setzen von initialen Konstanten.

Migration in eine partitionierte Tabelle oder eine IOT

Da im Schritt 1 immer das Anlegen der Tabelle vorgenommen wird, haben wir hier alle Freiheiten einer Redefinition. So kann ich an dieser Stelle eine partitionierte Tabelle definieren oder eine Index Organisierte Tabelle (IOT).
Bei der Übertragung in eine IOT muss beachtet werden, dass die Prozedur copy_table_dependents nicht oder nur bedingt verwendet werden kann, da der Primary Key Index und somit auch das Primary Key Constraint bereits definiert worden ist. Hier empfiehlt sich dann die granulare Methode über DBMS_REDEFINITION.REGISTER_DEPENDENT_OBJECT.

One Step Methode

Mit Oracle 12.1 ist eine Prozedur implementiert worden, über die die Reorganisation in einem Schritt durchgeführt werden kann. Dabei können einige Anpassungen vorgenommen werden

- Table Compression

- Index Compression
- Index Tablespace
- LOB Compression
- LOB Tablespace

Obwohl alle Parameter den Defaultwert NULL haben, muss mindestens ein Parameter beim Aufruf angegeben werden.

```
BEGIN
 DBMS_REDEFINITION.REDEF_TABLE(
   uname                        => 'schema'>,
    tname                        => '<table_orig>',
    index_tablespace             => '<index_tablespace>');
END;
/
```

Mit dieser Prozedur können recht einfach Reorganisationen durchgeführt werden, wenn der Zeitpunkt des FINISH unkritisch ist und auf besondere Aktionen wie Column Mapping oder Umwandlung in eine partitionierte Tabelle verzichtet werden kann.

Reorganisation einzelner oder mehrerer Partitionen

Einzelne Partitionen konnten schon immer mit dem Package reorganisiert werden, seit Oracle 12c können auch mehrere Partitionen auf einen Schlag reorganisiert werden. Hierfür muss für jede Partition eine normale Tabelle erzeugt werden, im `start_redef_table` muss dann bei den Parametern `int_table` und `part_name` jeweils eine Liste angegeben werden.

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE(
   uname        => '<schema>',
   orig_table   => '<part_table_orig>',
   int_table    => '<table_int_1>, <table_inte_2>, …',
   part_name    => '<part_orig_1>, <part_orig_2>, …')
END;
/
```

Reorganisation von Tabellen mit VPD Policies

Tabellen, auf denen VPD Policies liegen, können seit Oracle 12.1 ohne Probleme über das Package reorganisiert werden. Hierfür ist das neue Parameter `copy_vpd_opt` in der Prozedur `start_redef_table` eingeführt worden.

```
BEGIN
  DBMS_REDEFINITION.START_REDEF_TABLE (
    uname          => '<schema>',
    orig_table     => '<table_orig>',
    int_table      => '<table_int>',
    copy_vpd_opt   => DBMS_REDEFINITION.CONS_VPD_AUTO);
END;
/
```

Reorganisation von Tabellen mit Materialized Views
Tabellen auf denen Materialized Views liegen, können mit diesem package reorganisiert werden. Allerdings ist ein Fast Refresh während der Reorganisation erst seit Oracle 12.2 möglich. Hierfür ist der neue Parameter refresh_dep_mviews in den Prozeduren start_redef_table und redef_table eingeführt worden.

```
BEGIN
  DBMS_REDEFINITION.REDEF_TABLE(
    uname=>'<schema>',
    tname=>'<table_orig>',
    index_tablespace=>'<index_tablespace',
    refresh_dep_mviews=>'Y');
END;
/
```

Wiederaufsetzen nach Fehlern (Restart)
Wenn während des Reorganisationsprozesses ein Fehler aufgetreten ist, musste man den Prozess immer mit `abort_redef_table` abbrechen und neu starten. Seit Oracle 12.2 ist der Prozess nach der Fehlerkorrektur unter bestimmten Umständen wieder aufsetzbar. Über die View `dba_redefinition_status` kann man sehen, ob der Prozess wieder aufsetzbar ist (Spalte `RESTARTABLE`).

Möglichkeit eines Rollbacks
Mit Orale 12.2 hat man die Möglichkeit den Prozess nach erfolgreichem Abschluss komplett zurückzurollen. Beim Starten der Reorganisation muss diese Möglichkeit mitgegeben werden.

```
BEGIN
  dbms_redefinition.start_redef_table(
    '<schema>',
    'table_orig>',
    '<table_int>',
    enable_rollback=>TRUE);
END;
/
```

Soll ein Rollback durchgeführt werden, dann muss man die neue Prozedur ROLLBACK nutzen.

```
BEGIN
  dbms_redefinition.rollback(
    '<schema>',
    '<table_orig>',
    'table_int');
END;
/
```

Wenn erkannt worden ist, dass kein Rollback notwendig ist, dann muss der Prozess endgültig über die Prozedur ABORT_ROLLBACK beendet werden.

```
BEGIN
  dbms_redefinition.abort_rollback(
```

```
        '<schema>',
        '<table_orig>',
        'table_int');
    END;
    /
```

Fazit

Steht eine reine Reorganisation einer Tabelle an, so werden in der Praxis andere Methoden verwendet. Sobald allerdings in die Struktur einer Tabelle eingegriffen wird oder die Migration ohne Unterbrechung der Applikation erfolgen soll, so landet man sehr schnell bei diesem Package.
Einen weiteren großen Vorteil des Packages sehe ich darin, dass der Administrator bei sehr großen Tabellen auf den Zeitpunkt des Abschlusses besser einwirken kann.
Ich habe das Package in vielen Projekten sehr erfolgreich eingesetzt.

.

Kontaktadresse:
Klaus Reimers
ORDIX AG
Karl Schurz Str 19a
D-33100 Paderborn

Telefon: +49 (0) 5251-10630
Fax: +49 (0) 180-1673490
E-Mail info@ordix.de
Internet: www.ordix.de