Reorganisation – warum und wie?

Klaus Reimers
ORDIX AG
Paderborn

Schlüsselworte

Reorganisation – CTAS – DataPump – table move – dbms_redefinition – segment shrink

Einleitung

„Ich muss am Wochenende Überstunden machen, da ich diverse Tabelle 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?

Gründe für eine eventuelle Reorganisation / Workarounds

Diverse Gründe können die Reorganisation einer Tabelle sinnvoll erscheinen lassen:

Blockfüllgrad

Der Blockfüllgrad ist das entscheidende Kriterium für eine Reorganisation.

Je weniger gut ein Block gefüllt ist, desto mehr Leerbereiche werden durch den Database Buffer der SGA transportiert. Dies führt somit auch zu mehr physikalischen IO’s. Ein Full Table Scan wird grundsätzlich bis zur High Water Mark durchgeführt. Die High Water Mark bezeichnet den größten Block in einem Segment, der je allokiert worden ist. Über die Spalte `blocks` der View `dba_tables` ist die High Water Mark ersichtlich. Dieses Phänomen tritt vor allem bei Tabellen auf, auf denen viele DML-Aktionen laufen, vor allem bei einem hohen Anteil von Inserts und Deletes.

Als Workaround dient an dieser Stelle klassischerweise die Erhöhung des Parameters `PCTUSED`, damit der Block wieder schneller auf der Freelist erscheint und somit freier Platz im Block schneller wieder gefüllt werden kann. Bei Verwendung von `local managed tablespaces` mit der Option `SEGMENT SPACE MANAGEMENT AUTO` entfallen die Parameter `FREELISTS`, `FREELIST GROUPS` und `PCTUSED`.

Oracle baut je Segment ein indiziertes Bitmap auf, über welche die Belegung der einzelnen Blöcke verwaltet wird. Über diese Funktionalität wird der Platz sehr viel effizienter neu belegt.

Soll eine Tabelle komplett geleert werden, sollte dies mit `TRUNCATE` geschehen. Hierbei wird die High Water Mark auf den ersten Block gesetzt und alle Extents, außer dem initialen Extent, an das System zurückgegeben.

Musste per `DELETE` gelöscht werden, gibt es keinen Workaround.

Migrated / Chained Rows

Hierunter fallen folgende Aspekte:

- Sätze, die größer sind als ein Datenblock aufnehmen kann
- Sätze, die aufgrund eines Update nicht mehr ausreichend Platz im ursprünglichen Block finden (migrated row)

Ist die durchschnittliche Satzlänge größer als die Blockgröße, muss man die Frage stellen, ob die Blockgröße falsch gewählt wurde. Dieses Manko kann man nur durch ein Neuaufsetzen der Datenbank mit einer größeren Blockgröße (`db_block_size`) oder durch Verwendung eines Non-Standard-Tablespaces (ab 9i) versuchen zu beheben.

Liegen migrated rows vor, sollte zunächst eruiert werden, welche Sätze davon betroffen sind. Über das Kommando `analyze table ... list chained rows ...` kann man sich die ROWID's der betroffenen Sätze ausgeben lassen. Erfahrungen zeigen, dass in den meisten Systemen zu 90% nur auf aktuelle Daten zugegriffen wird. Wird vorwiegend auf aktuell eingegebene Datensätze zugegriffen, sind die chained rows nicht relevant. Mittels s`tatspack/AWR` kann man Aufschluss über die Zugriffe erhalten, welche über chained rows erfolgten. Dies allerdings nur global, nicht auf einzelne Tabellen bezogen.

Das Phänomen von chained (migrated) rows tritt auf, wenn Sätze sich durch Updates zu stark ausdehnen. Als Workaround sollte man zunächst `PCTFREE` erhöhen, um mehr Platz in den neu allokierten Blöcken frei zu lassen. Neue Sätze werden dann mit mehr Update-Spielraum gespeichert. Dies kann dazu führen, dass sich das Problem nach einer gewissen Zeit von selbst löst.

Methoden

CTAS

Hier erfolgt eine Reorganisation innerhalb der Datenbank.

Folgende Vorgehensweise muss gewählt werden:

1. exklusiven Zugriff auf der zu reorganisierenden Tabelle erzeugen
2. Tabelle auf neuen Namen unter Veränderung der Storage-Parameter kopieren
3. eventuell Foreign Keys ausschalten, die auf die Tabelle zeigen
4. Originaltabelle per `TRUNCATE` löschen
5. Daten per `INSERT ... SELECT` zurückschieben
6. eventuell Foreign Keys wieder einschalten, die auf die Tabelle zeigen

Die Reorganisation muss hier innerhalb eines Wartungsfensters durchgeführt werden.

DataPump

Folgende Vorgehensweise muss gewählt werden:

1. exklusiven Zugriff auf der zu reorganisierenden Tabelle erzeugen
2. Exportieren der zu reorganisierenden Tabelle
3. Aushängen aller Foreign Key Constraints, die auf diese Tabelle zeigen
7. Löschen der Tabelle per `TRUNCATE`
4. Importieren des generierten Export Files
5. 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 der zu reorganisierenden Tabelle:

- die Reorganisation wird gestartet über:

```
alter table <table_name>
      move tablespace <tablespace_name>
```

```
storage (...);
```

- Rebuild aller abhängigen Indexes 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 Indexe müssen ein Rebuild erfahren.

Auch bei Verwendung dieser Methode ist ein Wartungsfenster notwendig.

dbms_redefinition

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

Folgende Vorgehensweise muss gewählt werden:

1. Erstellen einer neuen zunächst leeren Tabelle
2. Starten der Reorganisation mittels
```
dbms_redefinition.start_redef_table(´<schema>´,
´<table>´,´<table_new>´);
```
3. Erzeugen aller notwendigen Objekte und Eigenschaften auf dieser Tabelle
```
dbms_redefinition.copy_table_dependents
```
4. Beenden der Reorganisation mittels
```
dbms_redefinition.finish_redef_table(´<schema>´,
´<table>´,´<table_new>´);
```
5. Löschen der Interimstabelle inklusive abhängiger 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.

Segment Shrink

Seit Oracle 9i kann man Tablespaces mit der zusätzlichen Eigenschaft ASSM (Automatic Segment Space Management) definieren. 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 tab ENABLE ROW MOVEMENT;
```

Über das Kommando
```
ALTER TABLE tab SHRINK SPACE COMPACT;
```

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

Über das Kommando
```
ALTER TABLE tab SHRINK SPACE CASCADE;
```

werden die Tabellen auch verdichtet, aber zusätzlich wird die High Water Mark nach vorne verschoben und eventuell freie Extents werden zurückgegeben.

Das Verschieben erfolgt intern über Insert/Delete. Die Indizes werden somit gepflegt und sind anschließend USABLE. Trigger werden nicht gefeuert.

Index-Reorganisation

Ein Index wird physikalisch in der Regel in Form eines B*Tree abgelegt. Allerdings kann der Baum durch starke Änderungen sehr löchrig werden. Es werden also viel mehr Blöcke im Baum belegt, als eigentlich notwendig wären.

Das Problem schlecht gefüllter Indizes tritt häufig auf, wenn die Spalte über eine Sequenz mit Daten versorgt wird. Dann ist jeder neue Satz ein neues Maximum im Index. Wenn dann auch noch ältere Sätze gelöscht werden, entstehen Lücken, die nie mehr gefüllt werden können.

Bei einer sehr fluktuativen Tabelle kann es zudem zu schlecht gefüllten Index-Blöcken kommen. Häufige Inserts können zum Block-Split führen.

Die Belegung innerhalb eines Indexes kann über den Befehl

```
alter index ... validate structure
```

analysiert werden.

Bei einem Index Rebuild wird der Baum komplett neu in anderen Extents aufgebaut. Während des Rebuild wird allerdings doppelter Speicherplatz benötigt. Über die Klausel `ONLINE` kann der Index auch nahezu ohne DML-Lock neu aufgebaut werden.

Beim Index Coalesce wird die Leaf-Block-Ebene des Indexes zusammengezogen. Es wird zur Laufzeit kein zusätzlicher Speicherplatz benötigt. Während des Coalesce wird kein DML-Lock gesetzt. Der Baum wird allerdings nicht neu aufgebaut, daher ist diese Methode nicht ganz so effektiv wie ein Rebuild.

Kontaktadresse:

Klaus Reimers
ORDIX AG
Westernmauer 12-16
D-33098 Paderborn

Telefon: +49 (0) 5251 1063-0
Fax: +49 (0) 180 1673490
E-Mail info@ordix.de
Internet: www.ordix.de