Oracle Datenbank Reorganisationen

Johannes Ahrends
Quest Software GmbH
Köln

Schlüsselworte:

Oracle Datenbank Administration, Online ReorganisationFragmentierung, Chained Row, Migrated Row, Index Rebuild, Redefinition, Locally Managed Tablespaces, Automatic Segment Space Management, ASSM, ASM

Einleitung

Im Oracle-Umfeld wird das Thema Reorganisation von Objekten (Tabellen und Indizes) immer wieder kontrovers diskutiert. Der Vortrag klärt, warum überhaupt reorganisiert werden muss und zeigt Möglichkeiten auf, mit unterschiedlichen Werkzeugen eine Reorganisation im laufenden Betrieb durchzuführen.

Warum muss reorganisiert werden?

Im Rahmen der Weiterentwicklung des Datenbank-Kerns hat Oracle in den vergangenen Jahrzehnten auch das Thema Reorganisation immer wieder aufgegriffen. Aus heutiger Sicht stellt sich die Frage, ob Reorganisation überhaupt noch erforderlich ist. Extentgrößen zur Optimierung der I/O Aktionen spielen keine Rolle mehr, Stripping und Mirroring sorgen für eine optimale Verteilung der Daten, Locally Managed Tablespaces erlauben beliebig viele Extents, Automatic Segment Space Management füllt Blöcke frühzeitig wieder aus und mit dem Oracle eigenen Filesystem ASM können alle Dateien automatisch verwaltet werden.

Dennoch gibt es immer noch einen Reorganisationsbedarf aus folgenden Gründen:

- Schachtelungstiefe des Index
- Migrated Rows
- Archivierung und damit Löschen von Tabelleninhalten
- Umstellung auf ein anderes Layout

Schachtelungstiefe des Index

Aus Performance-Sicht ist die Reorganisation von Indizes sicherlich die erfolgversprechendste, denn je kompakter ein Index aufgebaut ist, umso schneller können die einzelnen Datensätze gefunden werden. Bei Index-Zugriffen handelt es sich oft um das Lesen einzelner Blöcke und je mehr Sätze in einem Block referenziert werden, umso weniger I/Os sind nötig.

Speziell Indizes auf Nummernfelder (z.B. Primary Key) und solche auf Datumsfelder profitieren von einer regelmäßigen Reorganisation. Der Grund dafür liegt in der Struktur der Indizes. Normale Indizes (B*Tree-Indizes) weisen immer eine ausbalancierte Struktur auf. Da aber Nummernfelder für den Primärschlüssel in der Regel aufsteigend vergeben und beim Datum meist neue Werte am Ende eingefügt werden (außer bei Geburtstagen), würden diese Indizes sich eigentlich verschieben, das heißt, der Block mit den höchsten Zahlen bzw. neuesten Datumswerten würde immer wieder geschrieben. Das führt dazu, dass in den Blöcken mit den kleinsten Werten kaum Sätze referenziert

werden und in denen mit den größten Werten immer wieder ein Split durchgeführt wird. Zwar kann dies durch ein so genanntes „Reverse Key Indexing“ vermieden werden. In der Praxis ist dieses Vorgehen jedoch meist nicht brauchbar, da dann keine Range-Scans mehr möglich sind. Gerade beim Datumsfeld ist das aber der am häufigsten verwendete Zugriff.

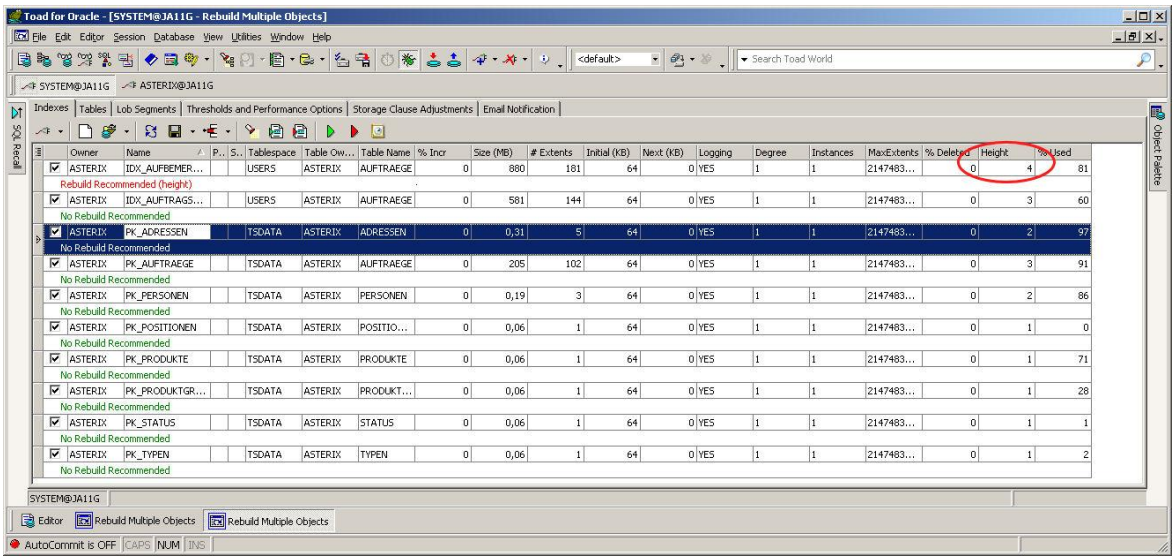

Abb. 1: Index Reorganisation

Migrated Rows

Die Problematik wird oft auch als Row Chaining bezeichnet, so auch, wenn entsprechende Tabellen analysiert werden. Allerdings kann es sich beim Row Chaining, das heißt beim Verketten von Sätzen über mehrere Oracle-Blöcke, auch um ein gewolltes Verhalten handeln. So ist Row Chaining beispielsweise erwünscht, wenn ein Datensatz so aufgebaut ist, dass er nicht in einen Block passen kann. Denn vor allen Dingen bei der Verwendung der Datentypen LONG bzw. LONG RAW ist es oft der Fall, dass allein dieses Feld schon größer ist als ein Oracle-Block (also heute in der Regel 8 KB). Dann muss der gesamte Datensatz über mehr als einen Block verteilt werden. Ebenso nützlich ist Row Chaining, wenn eine Tabelle mit mehr als 254 Spalten vorliegt: Oracle teilt diese Tabelle dann automatisch auf mehrere Blöcke auf. In diesem Fall ist jede Spalte − unabhängig davon, wie lang ein einzelner Satz ist − über mehrere Blöcke verteilt.

Von Migrated Rows spricht man, wenn in einem Datenbank-Block nicht mehr ausreichend Platz für die Erweiterung eines Datensatzes ist. Der Datensatz wird dann in einen neuen Block verschoben (migriert). Im alten Block verbleibt nur die Row-ID, die auf die neue Row-ID verweist. Für den Zugriff auf den Datensatz über einen Index bedeutet dies, dass zunächst der Block gelesen wird, in dem die alte Row-ID steht (dieser ist im Index eingetragen) und dann auf den neuen Block verwiesen wird. Die Anzahl der I/Os nimmt also mit zunehmender Anzahl von Migrated Rows zu. Bei normalen Änderungen durch Updates ist die Gefahr dieser Art von Row Chaining sehr gering, da standardmäßig über den Parameter PCTFREE 10 Prozent eines Blockes für Erweiterungen der Datensätze freigehalten werden. Bei gravierenden Änderungen der Struktur, wie sie beispielsweise durch Hinzufügen einer Spalte mit einem vordefinierten Wert, oder durch Massenupdates, bei denen z. B. eine Spalte, die vorher leer war, mit einem Wert gefüllt wird, entstehen, kommt es in der Regel immer zu Row Chaining. Daher sollte in einem solchen Fall immer daran gedacht werden, die Tabelle anschließend zu reorganisieren.

Archivierung

Die Archivierung von Datensätzen ist nach wie vor der häufigste Grund für eine Reorganisation. In einem solchen Fall werden ältere Datensätze, die einen bestimmten Status haben (z. B. Informationen über Aufträge, die älter als 2 Jahre sind und bereits geliefert sowie bezahlt wurden), aus der Datenbank herausgeschrieben und beispielsweise in einer einfachen Datei oder als Datenbank-Export abgelegt. Anschließend werden sie in der Datenbank gelöscht. Das Automatic Segment Space Management (ASSM) sorgt dafür, dass die entstandenen Lücken durch neue Datensätze wieder gefüllt werden. Allerdings kann dies bei wichtigen Tabellen zu massiven Performance-Problemen führen.

Strukturänderungen

Als letztes sei noch die pragmatische Notwendigkeit von Reorganisationen erwähnt: Strukturänderungen, wie zum Beispiel die Verlagerung von Objekten auf ein anderes Storage oder ASM, Ändern des Tablespace Layouts (Umstellung auf ASSM) oder einfach die Notwendigkeit, bestimmte Objekte in einen anderen Tablespace zu verlagern - auch dies ist eine Reorganisation.

Wie sollte reorganisiert werden?

Eine Reorganisation geht immer mit einer Einschränkung des Betriebs einher. Deshalb sollte bereits beim Datenbank-Design überlegt werden, welche Tabellen anfällig für Reorganisationen sind. Dies sind in der Regel die wirklich großen Tabellen. Daher empfiehlt es sich, diese Tabellen (und die zugehörigen Indizes) in separaten Tablespaces zu verwalten. Bei einer Reorganisation ergibt sich dann ein 1:1-Verhältnis zwischen Tabelle bzw. Index und Tablespace, was auch bedeutet, dass eigentlich der Tablespace reorganisiert wird.

Die Art der Reorganisation richtet sich vor allen Dingen nach der möglichen Auszeit für die Anwendung. Muss kein 7x24-Stunden-Betrieb sichergestellt werden, kann man leicht entsprechende Jobs aufsetzen, die die Objekte an Wochenenden oder während der Nachtzeiten reorganisieren. Dabei kommen dann die Oracle-Befehle `REBUILD INDEX` für die Index-Reorganisation oder `MOVE TABLESPACE` für die Tabellen-Reorganisation zum Einsatz. Sie sind einfach zu handhaben und die Fehlerquellen sind gering. Export/Import-Prozeduren sind im Gegensatz dazu auf keinen Fall empfehlenswert, da hier die große Gefahr besteht, außerhalb der Transaktionssicherheit eventuell sogar Daten zu verlieren.

Online-Reorganisation

Das bedeutet, dass während der Reorganisation die Objekte möglichst ohne Einschränkung zur Verfügung stehen. Ganz ohne Einschränkung wird die Reorganisation jedoch definitiv nicht durchzuführen sein. Durch das Kopieren der Objekte wird zusätzliches I/O erzeugt und, da in der Datenbank reorganisiert wird, werden entsprechende Redo- und Undo-Informationen mit generiert. Daher sollte zumindest eine lastarme Zeit gewählt werden.

Eine weitere Einschränkung stellt der Switch dar. Es kann erforderlich sein, beim Umschalten zwischen alter und neuer Tabelle eine kurze Tabellensperre zu setzen. Eine Anwendung, die quasi gleichzeitig einen Datensatz sperren möchte, wird dann eine Fehlermeldung als Antwort bekommen. Im Gegensatz dazu kann die Tabellensperre nicht gesetzt werden, wenn auf der Tabelle permanent Änderungen durchgeführt werden.

Außerdem ist zu beachten, dass bei derartigen Änderungen je nach Oracle-Version (Oracle 11g ist hier durch das so genannte „*Fine Grain Dependancy Tracking*“ wesentlich unempfindlicher als ältere Versionen) abhängige Objekte wie Views, PL/SQL-Prozeduren und Trigger ungültig werden und

entsprechend neu kompiliert werden müssen. Dies geschieht in der Regel automatisch bei der nächsten Ausführung. Handelt es sich aber um sehr komplexe Abhängigkeiten, könnte ein manueller Eingriff nötig sein.

	Oracle 10g	**Oracle 11g**
MOVE TABLESPACE	PL/SQL VALID Views VALID Index UNUSABLE	PL/SQL VALID Views VALID Index UNUSABLE
ENABLE ROW MOVEMENT	PL/SQL INVALID Views INVALID	PL/SQL VALID Views VALID
SHRINK SPACE	PL/SQL VALID Views VALID	PL/SQL VALID Views VALID
REDEFINITION (FINISH)	PL/SQL INVALID Views INVALID	PL/SQL VALID Views INVALID

Abb. 1: Fine Grain Dependancy Tracking

Index Reorganisation

Der oftmals effektivste Ansatz für eine Reorganisation besteht darin, den Index neu aufzubauen. Hierfür gibt es einen einfachen Befehl:

```
SQL> ALTER INDEX <indexname> REBUILD ONLINE;
```

Alternativ kann dabei auch ein neuer Tablespace gewählt werden.

Online-Tabellen-Reorganisation

Die Online-Reorganisation von Tabellen ist wesentlich aufwendiger. Oracle stellt heute zwar eine Reihe von Mechanismen dafür zur Verfügung, es muss aber im Einzelfall geprüft werden, welche Einschränkungen es dabei gibt.

Der einfachste Befehl ist folgender:

```
SQL> ALTER TABLE <tabellenname> SHRINK SPACE;
```

Dieser Befehl löst etwas Ähnliches aus wie die Defragmentierung bei MS-Windows. Die Blöcke werden wieder bis zur Grenze (`PCTFREE`) aufgefüllt und die *High-Water-Mark* der Tabelle wird zurückgesetzt. Allerdings funktioniert dieses Verfahren erst ab Oracle 10g und nur dann, wenn es sich um einen Tablespace mit ASSM handelt. Außerdem muss für die Tabelle zunächst einmalig das so genannte *Row Movement* aktiviert werden:

```
SQL> ALTER TABLE <tabellenname> ENABLE ROW MOVEMENT;
```

Leider funktioniert dieser Befehl nicht, wenn die Tabellen LONG- bzw. LONG RAW-Spalten haben oder wenn *Function-Based-Indizes* auf die Tabelle existieren. Es gibt noch weitere, aber in der Regel unkritischere Fälle, in denen `SHRINK SPACE` nicht gelingt.

Ein Vorteil von `SHRINK SPACE` ist, dass es sich um die einzige *In Place*-Reorganisation handelt, das heißt, es wird für die Reorganisation kein zusätzlicher Plattenplatz benötigt. Allerdings kann genau dieser Punkt auch zu einem Problem werden. Es gibt von Oracle-Anwendern immer wieder

Beschwerden über so genannte Dead-Locks (siehe My Oracle Support Bug 6761624). Dies betrifft die Anwendung direkt und ist damit besonders kritisch. Es kann sogar zu einem Stillstand in der Anwendung kommen, obwohl genau das durch eine Online-Reorganisation vermieden werden sollte.

Komplexe Tabellenreorganisation

Für eine komplexe Reorganisation von Tabellen gibt es seit der Oracle-Version 9i eine Reihe von PL/SQL-Paketen unter der Bezeichnung *Online Table Redefinition* (nicht zu verwechseln mit der in Oracle 11g Release 2 eingeführten *Edition Based Redefinition*). Diese Funktionen können neben einer einfachen Reorganisation auch Attribute ändern, wie zum Beispiel Löschen von Spalten, Ändern von Partitionierungsinformationen, Verschieben von Tabellen in einen neuen Tablespace und ähnliches.

Eine wichtige Einschränkung bei der Verwendung von *Online Table Redefinition* ist in der Verwaltung von LONG- bzw. LONG RAW-Spalten zu sehen. Zwar ist es möglich, Tabellen mit diesen Datentypen zu reorganisieren, allerdings werden die LONG- in CLOB- und die LONG RAW- in BLOB-Datentypen umgewandelt. Das *Redefinition Package* baut auf Materialized Views auf, das bedeutet, wenn während der Reorganisation einer Tabelle sehr viele Änderungen (DML) stattfinden, wird die Performance der Anwendung merklich beeinträchtigt.

Quest Space Manager with LiveReorg

Unter der Kurzbezeichnung „*LiveReorg*“ gibt es von Quest ein alternatives Produkt für die Reorganisation von Tabellen und Indizes in unterschiedlichen Datenbank-Editionen und -Versionen. Mit LiveReorg können auch Tabellen, die LONG- bzw. LONG RAW-Spalten enthalten, ohne Einschränkung reorganisiert werden. Außerdem können z.B. in einem Schritt alle Indizes in einen und die Tabellen in einen anderen Tablespace verlagert werden. Hilfreich ist auch, dass der gesamte Reorganisationsverlauf überwacht werden kann und definiert werden kann, dass ein Switch (d.h. der Austausch von Quell- und Zieltabelle) nur zu bestimmten Zeiten stattfinden darf.

Fazit

Es gibt immer noch die Notwendigkeit für Reorganisationen. Allerdings sind die Gründe dafür heute andere als vor 10 Jahren. Im Einzelfall muss immer wieder neu entschieden werden, welchen Nutzen die Reorganisation bringt und welcher Aufwand bzw. welches Risiko dem entgegen steht.

Kontaktadresse:

Johannes Ahrends

Quest Software GmbH

Im Mediapark 4e

D-50670 Köln

Telefon:	+49 (221)-57774 - 166
Fax:	+49 (221)-57774 - 50
E-Mail	johannes.ahrends@quest.com
Internet:	www.quest.com