Compression

Klaus Reimers

ORDIX AG, Paderborn

Schlüsselwörter
Compression, Index, Bulk, Heat Map

Einleitung

Mit der Datenbankversion 9i (Release 2) hat Oracle die Komprimierung von Tabellen für Bulk Loads eingeführt. Hierdurch kann der Speicherbedarf für Datensätze bei einem DIRECT LOAD oder bei dem Befehl CREATE TABLE ... AS SELECT ... reduziert werden.

Diese Art der Datenkomprimierung eignet sich in hervorragender Weise für Data Warehouse Umgebungen, in denen der Großteil der Daten durch Batch-Jobs in die Datenbank gebracht wird.

In der Datenbankversion 11g wird das neue Feature OLTP Table Compression eingeführt. Die Möglichkeit, Daten zu komprimieren wurde hierbei auf alle DML-Operationen wie INSERT, UPDATE und DELETE ausgeweitet. Hierbei handelt es sich nicht um eine schlichte Erweiterung, sondern der verwendete Algorithmus ist komplett überarbeitet worden, um den Overhead bei Schreiboperationen zu reduzieren. Dadurch hat sich der Einsatzbereich auf die OLTP-Tabellen erweitert.

IOTs können seit Oracle 9i, Indizes bereits seit Oracle 8i, komprimiert werden.

Table Compression – Compression for Direct Load Operations

Table Compression wird in den beiden unterschiedlichen Formen „Compression for Direct Load Operations“ und „Compression for all Operations“ angeboten.

Mit der Oracle 9i R2 Enterprise-Version wurde die „Compression for Direct Load Operations“ eingeführt. Die Datenkompression erfolgt auf Blockebene. Beim Einfügen in die Tabelle wird nach mehrfach auftretenden Datenwerten innerhalb eines Blocks gesucht. Für diese wird in jedem Block ein eigener Bereich reserviert. In diesem Bereich liegt die so genannte Symboltabelle. Mehrfach auftretende Datenwerte werden ausschließlich in der Symboltabelle gespeichert. Im restlichen Platz innerhalb des Blocks werden die eigentlichen Datensätze gespeichert. Diese enthalten für die komprimierten Datenwerte Zeiger auf die Symboltabelle.

Die Platzersparnis ergibt sich aus der Verwendung der Zeiger, die weniger Speicherplatz als die Datenwerte benötigen.

Die wesentlichen Eigenschaften dieser Kompression auf Blockebene sind:

- Ein Datenblock enthält die vollständige Information über alle Datensätze innerhalb des Blocks. Für einen Zugriff auf die Daten sind keine weiteren Informationen notwendig.

- Für den Anwender gibt es keine Änderungen beim Zugriff auf die Daten. Die Kompression ist vollständig unsichtbar.
- In komprimierter Form enthalten die Blöcke mehr Datensätze. Dadurch sinkt die I/O-Last.
- Die Daten werden in komprimierter Form im Database-Buffer gehalten. Es können also mehr Datensätze im zur Verfügung stehenden Puffer gehalten werden.

Um eine komprimierte Tabelle zu erstellen, wird beim Anlegen die Option COMPRESS angegeben. Damit wird die Eigenschaft COMPRESSION der Tabelle auf ENABLED gesetzt. Sehr wichtig ist auch, dass der Speicherparameter PCTFREE automatisch den Wert „0“ erhält. Es wird also kein Puffer für eventuelle Updates gelassen. Dies ist jedoch auch nicht notwendig, da Updates dem Konzept dieser Methode widersprechen.

Nachträgliche Änderungen der Daten einer komprimierten Tabelle führen zunächst zu einer Dekomprimierung. Nach dieser Änderung werden die Datensätze dann in unkomprimierter Form wieder eingefügt. Dadurch geht die angestrebte Platzersparnis also verloren.

Die Datenkompression erfolgt ausschließlich beim Einfügen von Daten in die Tabelle. Das beschriebene Verfahren wird jedoch nur angewandt, wenn mehrere Datensätze in einer Aktion eingefügt werden. Bei Datenmengen, die kleiner als ein Block sind, wird keine Kompression durchgeführt. Zum Einfügen muss eine der folgenden Blockoperationen (BULK LOAD oder BULK INSERT) verwendet werden:

- Verwendung des SQL*Loader mit der Option "direct = true"
- Verwendung des Befehls "CREATE TABLE . . . AS SELECT . . . "
- Einfügen von Daten mit "INSERT /*+ APPEND */ . . . "
- Einfügen von Daten mit "INSERT /*+ PARALLEL(tab_name, n) */ . . . "

Die Einsparungsquote ist stark davon anhängig, auf welche Weise die Daten geladen werden. Es sollte dafür gesorgt werden, dass die Daten in sortierter Reihenfolge in die Tabelle gelangen, um die Blöcke direkt optimal zu befüllen.

Table Compression – Compression for all Operations

Diese Kompressionsmethode steht kostenpflichtig ab Oracle 11g zur Verfügung.

Die grundlegende Methode der Komprimierung wurde hierbei nicht verändert. Die Einsparungsquote ist somit auch hier stark von der Gleichartigkeit der Datensätze innerhalb eines Datenblocks abhängig. So wird ein CREATE TABLE AS SELECT mit einer zusätzlichen Sortierung (ORDER BY) ein stärkeres Einsparungspotenzial bieten als eine im freien Dialog geänderte Tabelle.

Um den durch die Komprimierung der Daten entstehenden Overhead zu reduzieren, werden die Datensätze nicht einzeln, sondern blockweise in eine Art Batch-Job komprimiert.

Ein leerer Datenbankblock wird zunächst mit unkomprimierten Daten gefüllt, bis die Freispeichergrenze PCTFREE erreicht wird. Wird diese Grenze überschritten, wird der gesamte Inhalt des Datenbankblocks nach der oben genannten Kompressionsmethode komprimiert. Aus einem unkomprimierten Datenblock wird in diesem Schritt ein komprimierter Datenblock. Dieser

Datenbankblock steht weiterhin für die Aufnahme von Datensätzen durch INSERT-Statements bereit. Durch die weitere Aufnahme von Datensätzen wird aus dem unkomprimierten Datenblock ein teilweise komprimierter Datenblock. Durch erneutes Erreichen der Freispeichergrenze wird der teilweise komprimierte Datenbankblock nochmals komprimiert. Es entsteht wiederum ein vollständig komprimierter Datenblock. Diese Prozedur wird fortgesetzt, bis der Datenbankblock gefüllt ist.

Aufgrund dieser Vorgehensweise lösen nur wenige SQL-Statements die eigentliche Komprimierung der Daten aus, wodurch die Performance nur minimal beeinträchtigt wird.

Index Compression

Doppelte Indexwerte werden bei einem komprimierten Index nur einmalig gespeichert. Dadurch kann der Index-Baum kleiner und die Anzahl der Leaf-Blöcke minimiert werden.

Über den Befehl ALTER INDEX <index_name> VALIDATE STRUCTURE kann analysiert werden, ob eine Komprimierung eines Indexes lohnen könnte. Das Ergebnis dieser Analyse ist über die Views INDEX_STATS und INDEX_HISTOGRAM einsehbar.

Ein weiteres Ziel der Komprimierung ist eine Minimierung der Plattenzugriffe.

Eine Index-Komprimierung kann sehr sinnvoll sein, wenn die indizierte Spalte wenig unterschiedliche Werte enthält, oder wenn der Index aus mehreren Spalten zusammengesetzt ist.

Mit Oracle 12c ist eine weitere Methode hinzugekommen, die Index Advanced Expression. Dieses Verfahren ist kostenpflichtig und ist für sehr ungleichmäßig verteilte Indizes gedacht. Die Erfahrungen aus der Praxis sind bisher nicht vielversprechend.

LOB Compression

Mit Oracle 12c sind die SecureFiles die Standardimplementierung der LOBs geworden.

SecureFiles können komprimiert, dedupliziert und verschlüsselt werden.

Die Komprimierung basiert auf dem klassischen zip-Verfahren, bei der Deduplizierung werden identische LOBs lediglich einmal gespeichert.

In Projekten haben wir bei der Verwendung dieser Methodiken starke Einsparungen erreicht.

rman Compression

Hier bietet Oracle vier verschiedene Komprimierungsverfahren an. Die Basic-Methode ist schon seit Oracle 8i verfügbar und ist nach wie vor kostenfrei. Alle anderen Verfahren sind mit Oracle 11g eingeführt worden und erfordern die Compression-Option.

Die Laufzeiten und Komprimierraten sind dabei sehr unterschiedlich.

DataPump Compression

Die verwendeten Verfahren sind identisch zur rman Compression. Allerdings sind beim DataPump alle Komprimierungsmethoden kostenpflichtig.

Ausblick auf den Vortrag

Im Vortrag werden die einzelnen Verfahren mit Hilfe anschaulicher Grafiken technisch erklärt. Mitlaufende Skripte verdeutlichen die einzelnen Wirkungsweisen. Besonderen Wert wird dabei auf die Einsatzmöglichkeiten in der Praxis gelegt, Performance-Betrachtungen spielen dabei selbstverständlich eine große Rolle.

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

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