Datenmengen explodieren und stellen die Betreiber von Datenbanken vor große Herausforderungen. Dieser Artikel gibt einen Überblick über die verschiedenen Einsatzgebiete und Techniken zur Verdichtung von Daten in einer Oracle-Datenbank.

Sind wir eigentlich ganz dicht?

Abbildung: Fotolia

Eero Mattila, Quest Software GmbH

Im Laufe der Jahre hat Oracle unterschiedliche Möglichkeiten zur Verdichtung von Daten eingeführt, angefangen mit der Index-Key-Komprimierung in der Version 8.1.5 über einfache Tabellen-Komprimierung in 9.2, Backup-Komprimierung in 10*g*, erweiterte Komprimierung in 11*g* Release 1 bis hin zur hybriden Spaltenkomprimierung im Zusammenhang mit Exadata in 11*g* Release 2. Diese Techniken zur Verdichtung stellen einerseits Chancen zur Speicherplatz-Einsparung und Performance-Vorteile dar, können aber auch Zusatzkosten – monetär wie performancebedingt – verursachen.

Datenexplosion und ihre Folgen

Das kollektive Wissen der Menschheit nimmt täglich zu und wird immer eifriger nicht nur dokumentiert, sondern auch über verschiedenste Medien verbreitet. Gesetzliche Vorschriften verpflichten uns, bestimmte Daten sogar über Jahrzehnte aufzubewahren. Dies führt zu einem explosionsartigen Wachstum des Bedarfs an Speicherplatz, der wiederum schlicht Geld kostet. Hinzu kommt, dass die Verarbeitung immer größerer Datenmengen auch mehr Zeit kostet – die Performance der Anwendungen nimmt ab. Das grundsätzliche Problem – die Datenexplosion an sich – lässt sich nicht ändern, also müssen wir andere Möglichkeiten finden, damit umzugehen. Komprimierung von Daten an verschiedenen Stellen und auf unterschiedliche Arten und Weisen kann oft gute Dienste zu diesem Zweck leisten: Sie hilft einerseits, Speicherplatz zu sparen, andererseits aber auch, Zugriffszeiten auf Datenbank-Inhalte zu optimieren.

```
CREATE INDEX ON tabelle (spalte1, spalte23, …) COMPRESS x;
ALTER INDEX index REBUILD COMPRESS x;
```

Listing 1

```
SELECT blocks, opt_cmpr_pctsave, opt_cmpr_count
FROM index_stats;

    BLOCKS OPT_CMPR_COUNT OPT_CMPR_PCTSAVE
---------- -------------- ----------------
       988              2               37
```

Listing 2

Index Key Compression

Bereits in der Version 8.1.5 der Oracle-Datenbank wurde die Möglichkeit eingeführt, Indizes durch Komprimierung zu optimieren. Hierbei handelt es sich um ein Feature, das in jeder Edition der Datenbank zur Verfügung steht und nicht separat lizenziert werden muss. Bei der „Index Key Compression“ werden sich wiederholende Werte in führenden Spalten von Indizes dedupliziert, indem sie nur einmal im Blockheader gespeichert sind. Je nach Selektivität der zu komprimierenden Schlüssel kann der benötigte Speicherplatz des Index erheblich reduziert und dadurch deutlich geringerer I/O bei Index-Zugriffen erzielt werden. Neben „B*Tree“-Indizes ist die Komprimierung auch für „Index Organized Tables“ möglich. Die Komprimierung kann sowohl beim Erstellen eines neuen Index als auch nachträglich definiert werden (siehe Listing 1). Mit x wird die Anzahl der zu komprimierenden führenden Spalten angegeben.

Ob und wie viel ein Index komprimiert werden sollte, kann mit dem Befehl „ANALYZE INDEX index VALIDATE STRUCTURE;“ ermittelt werden. Anschließend gibt die View „INDEX_STATS“ Auskunft über die zu erzielende Ersparnis und die optimale Komprimierungs-Einstellung (siehe Listing 2).

Demnach empfiehlt sich in diesem Fall eine Komprimierung des Index mit „ALTER INDEX comp_test REBUILD COMPRESS 2;“, wobei

„COMPRESS 2“ den Wert in der Spalte „OPT_CMPR_COUNT“ betrifft. Es ist mit einer Reduzierung der Indexblöcke um ca. 37 Prozent zu rechnen. Listing 3 dokumentiert die erneute Abfrage gegen „INDEX_STATS“. Wie erwartet, hat sich die Anzahl der Blöcke in dem Index um ein gutes Drittel reduziert.

Bei „ANALYZE INDEX … VALIDATE STRUCTURE“ ist zu beachten, dass der Befehl einen DML-Lock auf den Index erzeugt, sodass insbesondere bei großen Indizes die Performance von Inserts, Updates und Deletes während des Analyze-Vorgangs beeinträchtigt werden kann.

Tabellen-Komprimierung

Oracle 9*i* Release 2 führte erstmals die Komprimierung für Tabellen-Daten ein, zunächst allerdings nur für ausgewählte, sogenannte „Bulk-Load-Operationen“ (Direct Path SQL*Load, CREATE TABLE AS SELECT, paralleles DML und INSERT mit APPEND-Hint). Dies bedeutete, dass mittels normaler DML-Befehle eingefügte oder geänderte Daten nicht komprimiert wurden. Somit war dieses Feature vor allem für Data-Warehouse-Umgebungen interessant. Allerdings gab (und gibt) es dort häufig Einschränkungen – sei es, dass der ETL-Vorgang keinen Direct Load ermöglicht oder dass ein CTAS aufgrund des Datenvolumens zu viel Zeit in Anspruch nehmen würde.

Mit der Version 11*g* kam die (separat zu lizenzierende) „Advanced Compression“-Option. Sie erweitert die Tabellen-Komprimierung auf INSERT- und UPDATE-Kommandos, sodass die Datenverdichtung auch in OLTP-Anwendungen zum Tragen kommt. Die oben beschriebene ursprüngliche Funktionalität für Bulk-Operationen heißt nun „Basic Compression“ und ist weiterhin ohne Zusatzkosten verwendbar.

Die Komprimierung der Tabellen-Daten erfolgt auf Block-Ebene: Wiederholte Spaltenwerte werden – wie bei der Index-Key-Komprimierung – einmalig in einer Symbol-Tabelle im Blockheader abgelegt und im Block selbst lediglich referenziert. Je nach Charakter der Daten können dadurch erheblich mehr Datensätze in einem Block untergebracht werden, was wiederum die Anzahl der physikalischen und logischen I/O-Operationen reduziert.

```
SELECT blocks, opt_cmpr_pctsave, opt_cmpr_count
FROM index_stats;

    BLOCKS OPT_CMPR_COUNT OPT_CMPR_PCTSAVE
---------- -------------- ----------------
       621              2                0
```

Listing 3

```
CREATE TABLE […] COMPRESS FOR OLTP;
```

Listing 4

```
CREATE TABLE doag_normal
AS SELECT * FROM dba_objects;

CREATE TABLE doag_compr COMPRESS FOR OLTP
AS SELECT * FROM dba_objects;

SELECT segment_name, blocks
FROM user_segments
WHERE segment_name LIKE 'DOAG%';
```

Listing 5

```
SEGMENT_NAME             BLOCKS
-------------------- ----------
DOAG_COMPR                  384
DOAG_NORMAL                1152
```

Listing 6

```
ALTER TABLE tabelle COMPRESS FOR OLTP;
```

Listing 7

```
ALTER TABLE tabelle MOVE COMPRESS FOR OLTP;
```

Listing 8

```
CREATE TABLESPACE […] DEFAULT COMPRESS FOR OLTP […];
ALTER TABLESPACE DEFAULT COMPRESS FOR OLTP;
```

Listing 9

Beim Erzeugen einer neuen Tabelle wird die Komprimierung durch die Klausel „COMPRESS FOR“ definiert, gefolgt von den Operationen, für die die Komprimierung gelten soll. Gültige Optionen sind „BASIC“ (nur Bulk Inserts und CTAS) sowie „OLTP“ (alle Operationen, siehe Listing 4). Bei Exadata gibt es noch weitere Optionen – dazu später mehr.

Bereits bei Tabellen recht übersichtlicher Größe lässt sich die Platzerspar-

nis feststellen. Im folgenden Beispiel werden zwei identische Tabellen erstellt, einmal komprimiert, einmal nicht. Anschließend wird die Anzahl der Blöcke anhand der View USER_SEGMENTS kontrolliert (siehe Listing 5). Das Ergebnis ist eindeutig (siehe Listing 6). Listing 7 zeigt, dass sich die Verdichtung auch nachträglich einstellen lässt.

Dabei ist jedoch zu beachten, dass bereits vorhandene, vollständig ausgefüllte Blöcke nicht automatisch komprimiert werden. Um diese Blöcke zu verdichten, müsste man zunächst Datensätze löschen. Wird die Tabelle jedoch gleichzeitig reorganisiert, greift die Komprimierung sofort (siehe Listing 8). Die Komprimierung kann außerdem als Standard-Einstellung für alle neuen Tabellen in einem Tablespace definiert werden, sowohl bei dessen Erstellung als auch nachträglich (siehe Listing 9).

Welche Tabellen man komprimieren sollte

Typische Kandidaten für eine Komprimierung sind natürlich große Tabellen, die sich wiederholende Daten enthalten. Um festzustellen, ob und wie stark sich eine Tabelle verdichten lässt, steht ab Oracle 11*g* Release 2 das mitgelieferte Package „DBMS_COMPRESSION“ zur Verfügung. Für ältere Versionen hält das Oracle Technology Network das Package „DBMS_COMP_ADVISOR“ zum Herunterladen bereit. Die Prozedur „GET_COMPRESSION_RATIO“ beziehungsweise „GETRATIO“ in der älteren Fassung berechnet für die angegebene Tabelle den zu erwartenden Komprimierungsgrad für die gewünschte Komprimierungsart. Derzeit beschränkt sich die Unterstützung auf BASIC, OLTP sowie die verschiedenen Varianten der Exadata Hybrid Columnar Compression (siehe weiter unten). Hinweise für Indizes oder unstrukturierte Daten gibt es leider nicht.

Was Index- oder Tabellen-Komprimierung kostet

Der reine Lizenzkosten-Aspekt der Advanced-Compression-Option soll hier nicht weiter vertieft werden. Man muss sich dessen bewusst sein, wie auch der Tatsache, dass die Funktionalität in jeder Installation einer Enterprise Edition vorhanden und ohne besondere Freischaltung einsetzbar ist. Sobald das Zauberwort „COMPRESS“ in einem „CREATE“- oder „ALTER“-Statement erscheint, ist eine Lizenzierung fällig. Wer die Option nicht besitzt und die Verwendung seinen Anwendern verwehren möchte, kann auf den nicht dokumentierten „init.ora“-Parameter „_OLTP_COMPRESSION=FALSE“ zurückgreifen – aber auf eigene Gefahr.

Komprimierung bedeutet selbstverständlich Umrechnung von Daten. Bedeutet das, dass wir uns Platzersparnis und I/O-Vorteile auf Kosten von CPU-Verbrauch erkaufen? Kann jedes „INSERT“ oder „UPDATE“ unmittelbar ein Umschreiben von Block-Headern und -Inhalten verursachen? Müssen bestehende Anwendungen angepasst oder darauf vorbereitet werden?

Für lesende Zugriffe bedeutet die Komprimierung – oder vielmehr Dekomprimierung – meist keinen messbaren Overhead. Dadurch, dass die Symbol-Tabelle auf Block-Ebene geführt wird, geschieht die Umrechnung extrem schnell. Die Lese-Performance lässt sich in vielen Fällen sogar erheblich steigern, weil weniger Blöcke gelesen werden müssen, sodass die I/O-Vorteile den ohnehin geringen CPU-Overhead deutlich überwiegen.

Interessanter ist die Frage nach den DML-Vorgängen in schreibintensiven OLTP-Systemen. Glücklicherweise lösen einzelne INSERTs oder DELETEs keine Komprimierungs-Vorgänge aus, sondern die Komprimierung findet asynchron statt, sobald ein bestimmter Füllgrad des Blocks erreicht wird. Umfangreiche UPDATEs können jedoch zu „Migrated Rows“ und damit zu Performance-Einbußen führen. Gründliche Tests sind daher in jedem Fall erforderlich, ehe bestehende Daten in Produktions-Umgebungen umgestellt werden können. Der Einsatz von Index- und Tabellenkomprimierung ist aus der Sicht der Datenbankanwendungen vollkommen transparent. Anpassungen bestehender oder die Vorbereitung neuer Applikationen ist nicht erforderlich.

Komprimierung unstrukturierter Daten

Besonders unstrukturierte Daten nehmen häufig sehr viel Platz in Anspruch und sind somit prädestinierte Kandidaten für eine Komprimierung, sofern sie nicht von vornherein in einem komprimierten Format vorliegen. In der Datenbank-Version 10*g* wurde das Package „UTL_COMPRESS“ vorgestellt, mit dem große binäre Objekte (RAW, BLOB und BFILE) komprimiert werden können.

„SecureFiles“ in 11*g* werden gern als die „nächste Generation von LOBs“ bezeichnet. Neben verschiedenen Vorteilen bezüglich Performance und Administration bieten sie – auch wieder unter Nutzung der Advanced-Compression-Option – erweiterte Möglichkeiten zur Datenverdichtung. Als „SecureFiles“ gespeicherte große Objekte können einerseits komprimiert werden, wobei die drei Komprimierungsstufen „LOW“, „MEDIUM“ und „HIGH“ zur Verfügung stehen. Zu beachten ist, dass die Komprimierung der LOBs unabhängig von der regulären Tabellen-Komprimierung ist: Die Tabellen-Komprimierung komprimiert keine LOBs und umgekehrt. Neben der Komprimierung lassen SecureFiles auch die Deduplizierung der Objekte zu: Identische Dateien werden einmal gespeichert und von weiteren Datensätzen referenziert. Diese beiden Methoden können zu drastischen Einsparungen an Speicherplatz führen.

Hybrid Columnar Compression

Die oben beschriebenen Tabellen-Komprimierungsarten sind unabhängig vom verwendeten Speichersystem einsetzbar. Die hybride Spalten-Komprimierung oder Hybrid Columnar Compression (HCC) setzt dagegen einen Exadata-Storage-Server beziehungsweise Oracle SAN oder NAS-Storage-Server voraus. HCC unterscheidet zwischen Data-Warehouse- und Archiv-Systemen und stellt für beide optimierte Komprimierungs-Algorithmen zur Verfügung.

Bei HCC handelt es sich um eine Kombination aus spalten- und zeilenbasierter Datenspeicherung. Die Datensätze werden in sogenannten „Logischen Komprimierungsgruppen“ or-

ganisiert, die mehrere Blöcke umfassen und die Daten spaltenweise in verschiedenen Blöcken abspeichern. Dadurch werden sich wiederholende Daten in den gleichen Blöcken konzentriert, sodass eine extrem effiziente Komprimierung möglich ist.

Warehouse-Anwendungen sind dadurch geprägt, dass extrem große Daten-Volumina mittels „SELECT“-Statements abgefragt und meist nie oder sehr selten verändert werden. Die Warehouse Compression verwendet einen speziell für Abfragen optimierten Algorithmus, wobei zwischen zwei Komprimierungsstufen gewählt werden kann. Typische Komprimierungsraten sind hier sechs bis zehn im Vergleich zu nicht komprimierten Daten.

Archivsysteme dienen zur Speicherung von historischen Daten, die aufgrund von gesetzlichen Vorschriften sogar jahrzehntelang vorgehalten werden müssen. Häufig werden solche Daten auf Band-Archiven abgelegt. Bei Bedarf müssen sie jedoch für den Zugriff wiederhergestellt werden, was sehr aufwändig sein kann. Die Archive Compression von HCC ermöglicht es, historische Daten auf Platzersparnis zu optimieren und trotzdem jederzeit im Zugriff zu behalten. Der Komprimierungsfaktor liegt bei Archive Compression häufig bei mindestens fünfzehn – aus einem Terabyte Daten werden dann rund 65 GB.

Backup-Komprimierung

Das Wachstum der Datenmengen führt zwangsläufig auch zu größeren und damit langsameren Daten-Sicherungen. Nicht überraschend, kann auch hier die Komprimierung wertvolle Vorteile verschaffen. Oracle Recovery Manager sichert Daten blockweise physikalisch und ermöglicht die Wiederherstellung einer Datenbank, eines Tablespace oder auch einzelner Datenblöcke. Für Exports oder logische Backups einzelner Schemata oder Objekte bietet sich Data Pump an. Beide Werkzeuge sind in der Lage, Daten unter Nutzung der Advanced-Compression-Option zu komprimieren, RMAN auch ohne das kostenpflichtige Add-on. Bei RMAN wird die Komprimierung mit einem Befehl eingeschaltet (siehe Listing 10). Anschließend kann der Komprimierungsalgorithmus ausgewählt werden (siehe Listing 11). Die Komprimierungsstufe „BASIC“ ist ohne die Advanced-Compression-Option verwendbar.

```
CONFIGURE DEVICE TYPE
[DISK|SBT] BACKUP TYPE TO COM-
PRESSED BACKUPSET;
```

Listing 10

```
CONFIGURE COMPRESSION ALGORITHM
'[BASIC|LOW|MEDIUM|HIGH]';
```

Listing 11

Da die Performance der verschiedenen Komprimierungsstufen stark von zahlreichen Faktoren (Charakter der Daten, Netzwerk-Konfiguration, CPU-Ressourcen etc.) abhängt, können keine allgemein gültigen Empfehlungen abgegeben werden. Höhere Komprimierung ist meist mit höherem CPU-Verbrauch verbunden, kann jedoch durch geringere Netzwerk-Belastung von Vorteil sein. Ausführliche Tests in der jeweiligen Umgebung sind letztendlich ausschlaggebend für die optimale Auswahl der Parameter. Besonders wichtig dabei ist, das Restore gründlich zu testen: Die kleinste Sicherung ist nichts wert, wenn die Wiederherstellung ewig dauert.

Data Pump Exports kennen ebenfalls den Parameter „COMPRESSION“, der folgende Werte akzeptiert: „ALL“, „METADATA_ONLY“ und „DATA_ONLY“. Sowohl RMAN als auch Data Pump dekomprimieren die Daten transparent beim Wiederherstellen beziehungsweise Importieren.

Fazit

Die Oracle-Datenbank bietet zahlreiche Mechanismen zur Komprimierung von Daten und dadurch zur Optimierung von Speicherplatz und Anwendungs-Performance. Während einzelne dieser Methoden ohne Extrakosten eingesetzt werden können, sind bei vielen verlockenden Funktionen zusätzliche Lizenzen fällig, sodass deren Einsatz wohlüberlegt werden muss. Performance ist ein wichtiger Aspekt bei allen Anwendungen, und die Komprimierung kann sich in gewissen Konstellationen auch negativ darauf auswirken. Daher ist ein umfangreiches Testen unbedingt notwendig, bevor Komprimierung produktiv eingesetzt wird. Wer aber mit stetig wachsenden Datenmengen, Performance-Verlust und limitierter Speicherkapazität zu kämpfen hat, tut gut daran, sich die verschiedenen Komprimierungsmöglichkeiten genauer anzuschauen.

Eero Mattila
eero.mattila@quest.com