Möglichkeiten und Grenzen von Oracle Advanced Compression

Olaf Herden
Duale Hochschule BW
Campus Horb, Florianstr. 15, 72160 Horb

Schlüsselworte:

Oracle, Datenbank, Oracle Advanced Compression, Kompression, Komprimierung.

1 Einleitung

Das Volumen von Datenbanken ist in ständigem Wachstum, was u.a. folgende Gründe hat:

- Natürliches Wachstum der Geschäftsdaten
- Migration existierender Desktop-Applikationen in unternehmensweite Anwendungen
- Entwicklung neuer Applikationen
- Gesetzliche Anforderungen (z.B. Basel II oder Sarbanes Oxley) Daten länger aufzubewahren
- Bedingt durch größere Bandbreiten nimmt die Verteilung und damit Speicherung umfangreicher unstrukturierter und multimedialer Daten zu
- Redundante Speicherung von Daten in Data Warehouses
- Neue Technologien wie RFID produzieren große Datenmengen

Zusammenfassend kann gesagt werden, dass der von Datenbanken benötigte Festplattenspeicherbedarf stets anwächst. Sowohl die Beschaffung als auch der Energieverbrauch während des Betriebs sind Kostenfaktoren im Rechenzentrum. Darüber hinaus verlangsamen sich mit steigendem Datenvolumen die Antwortzeiten für Anfragen, ebenso erhöht sich die für das Backup benötigte Zeit. Um diesen unerwünschten Effekten zu begegnen sind Kompressionstechnologien eine Möglichkeit.

Um den Effekt der Kompression messen zu können, wird als Maß Kompressionsfaktor verwendet [Sayo05]. Dieser ist als Quotient aus dem Platz für die unkomprimierte und dem Platz für die komprimierte Speicherung definiert. Liegen beispielweise ursprünglich Daten mit einem Volumen von 100 MB vor und diese lassen sich auf 20 MB komprimieren, dann ist der Kompressionsfaktor 5. Manchmal wird auch das explizite Verhältnis (im Beispiel 1:5) oder der eingesparte Speicherplatz (hier 80%) als Maß angegeben.

Die besten Kompressionsfaktoren lassen sich mit Kompressionsalgorithmen wie denen der Familie der Limpel Ziv Algorithmen erreichen (LZ77, LZ78, LZSS, LZW, LZMA, siehe z.B. ([ZiLe77], [Salo06], [Sayo05])). All diese Algorithmen erzeugen jedoch einen Overhead, der im Datenbankumfeld nicht gerechtfertigt wäre. Das Ziel der Kompression in Datenbanksystemen ist daher einerseits das Erreichen eines guten Kompressionsfaktors, andererseits darf der Overhead für die (De)kompression und für das Logging nicht aus dem Ruder laufen, ebenso sollten Einfüge- und Aktualisierungsoperationen nicht signifikant verlangsamt werden. Neben diesem generellen Problem können auch spezielle Probleme auftreten, z.B. wird in [Mann09] davon berichtet, dass in Oracle 10g durch Bind-Variable-Peeking für eine mit Bind-Variablen versehene Anfrage ein unpassender Ausführungsplan erzeugt werden kann.

Der weitere Beitrag ist folgendermaßen aufgebaut: Im folgenden Abschnitt wird das Konzept der Advanced Compression Option in Oracle 11g R2 vorgestellt. Anschließend werden einige mit der Kompression erreichte Resultate aus Fremdquellen und eigenen Untersuchungen vorgestellt. Eine Zusammenfassung und ein Ausblick beenden den Beitrag.

2 Oracle Advanced Compression

Die Advanced Compression Option in Oracle 11g R2 ist nicht der erste Kompressionsansatz von Oracle. In der Vergangenheit gab es bereits folgende Ansätze:

- *Kompression von Indexen* in Oracle 8i
- *Basic Table Compression* in Oracle 9i R2 ermöglichte per Bulk Load eingefügte Daten zu komprimieren
- *LOB Kompression* in Oracle 10g
- *OLTP Table Compression* in Oracle 11g R1 wendete Kompression bei allen Datenmanipulationsanweisungen an, wobei auf eine Minimierung des Overhead geachtet wurde, so dass diese Option auch für OLTP-Anwendungen mit der für sie typischen Änderungsdynamik verwendet werden kann

Die *Advanced Compression Option* ([CGH+08], [Nand08], [Orac09]) umfasst folgende Aspekte: Kompression von Sicherungsdateien, von SecureFile-Objekten als Nachfolger von LOBs (Large Objects) und als Hauptaspekt Kompression strukturierter Daten in Tabellen (entspricht der oben genannten OLTP Table Compression).

Kompressionstechnologie ist Bestandteil der Oracle Enterprise Edition. Während die Basic Table Compression Bestandteil der Oracle 11g EE ist, ist die Oracle Advanced Compression Option zusätzlich lizenzpflichtig.

2.1 Kompression von Backups

Oracle stellt zwei Werkzeuge zur Backup-Erstellung zur Verfügung: RMAN erzeugt physische Backups durch blockweises Kopieren der Datenbank, Datapump erzeugt ein logisches Backup durch Umspeichern der Datensätze in Dateien. Bei beiden Werkzeugen besteht die Möglichkeit eine Kompressionsoption einzustellen.

2.2 Kompression von SecureFile-Objekten

Häufig werden in Datenbanken exakte Kopien von Dateien abgelegt (siehe Abbildung 1 oberer Teil), z.B. in E-Mail-Systemen, wenn mehrere Benutzer den gleichen Anhang zugesendet bekommen. In diesem Szenario ist die Deduplikation eine sehr gute Möglichkeit zur Kompression. Hierbei wird, wie in Abbildung 1 im unteren Bereich dargestellt, von einem SecureFile-Objekt nur genau eine Instanz physisch gespeichert und von den anderen Datensätzen ein Verweis hierauf vorgenommen.
Neben der Speicherplatzersparnis beschleunigt SecureFile Deduplikation auch die Verarbeitung: Zum einen werden Schreib- und Kopieroperationen performanter, weil nur Verweise gesetzt werden müssen, zum anderen können Leseanfragen beschleunigt werden, wenn sich das entsprechende SecureFile-Objekt bereits im Puffer befindet.

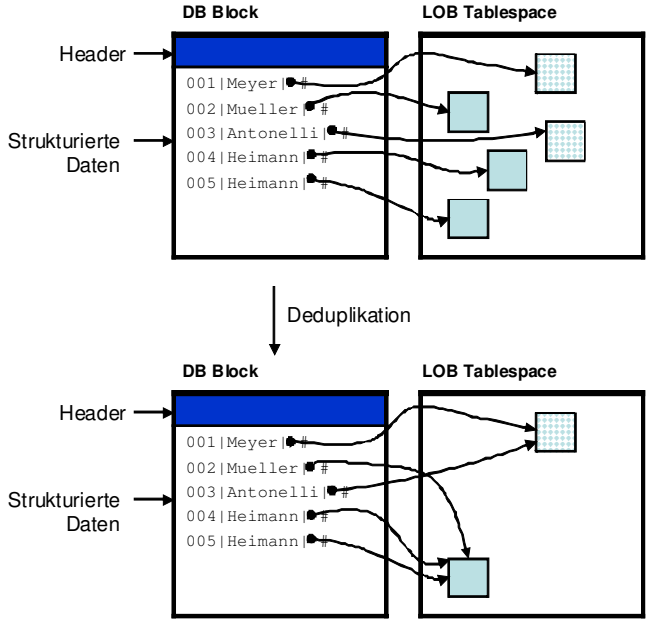

Abb. 1:Deduplikation von SecureFile-Objekten

Die Syntax für die Deduplikation von SecureFile-Objekten sieht folgendermaßen aus:

```
CREATE TABLE myTable(
    ...
  myObject BLOB
)
  LOB(myObject) STORE AS SECUREFILE
  (TABLESPACE LOB_TBS DEDUPLICATE);
```

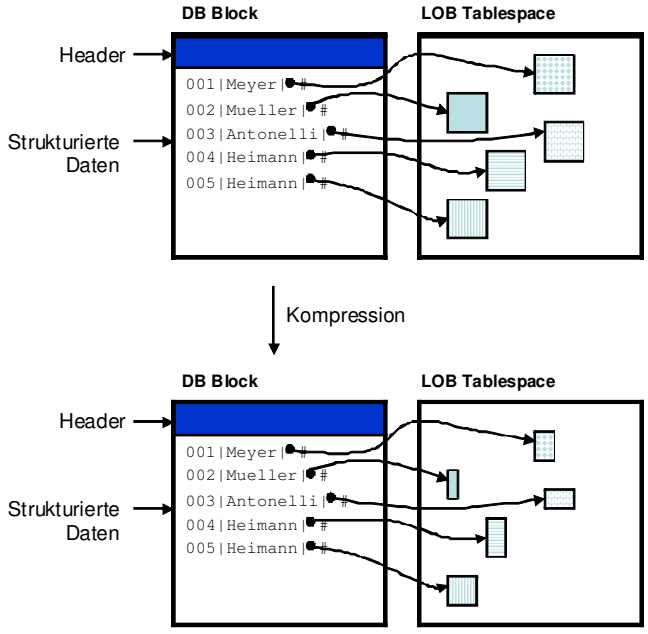

Abb. 2:Kompression von SecureFile-Objekten

In anderen Szenarien (z.B. Bilder oder Beschreibungen im XML-Format zu Produkten) liegen in der Regel keine exakten Kopien der SecureFile-Objekte vor und das zuvor beschriebene Vorgehen der Deduplizierung wäre wirkungslos. Aus diesem Grunde ist die Kompression von SecureFile-Objekten eine weitere Möglichkeit, bei der jedes einzelne SecureFile-Objekt komprimiert abgespeichert wird, was in Abbildung 2 dargestellt ist.

Das vorherige Überprüfen des Vorteils der Kompression durch die Datenbank verhindert unnötiges Komprimieren (und wieder Dekomprimieren).

Die Kompression von SecureFile-Objekten wird durch folgende Syntax erreicht:

```
CREATE TABLE myTable(
   ...
  myObject BLOB
)
  LOB(myObject) STORE AS SECUREFILE
  (TABLESPACE LOB_TBS COMPRESS);
```

Dem Schlüsselwort `COMPRESS` kann in der Anweisung `LOW`, `MEDIUM` (Standardwert) oder `HIGH` folgen, um anwendungsspezifisch zwischen Kompressionsgrad und aufzubringender CPU-Last abzuwägen. In der genannten Reihenfolge wächst Kompressionsfaktor, gleichzeitig steigt aber auch die CPU-Last.

2.3 Kompression strukturierter Daten

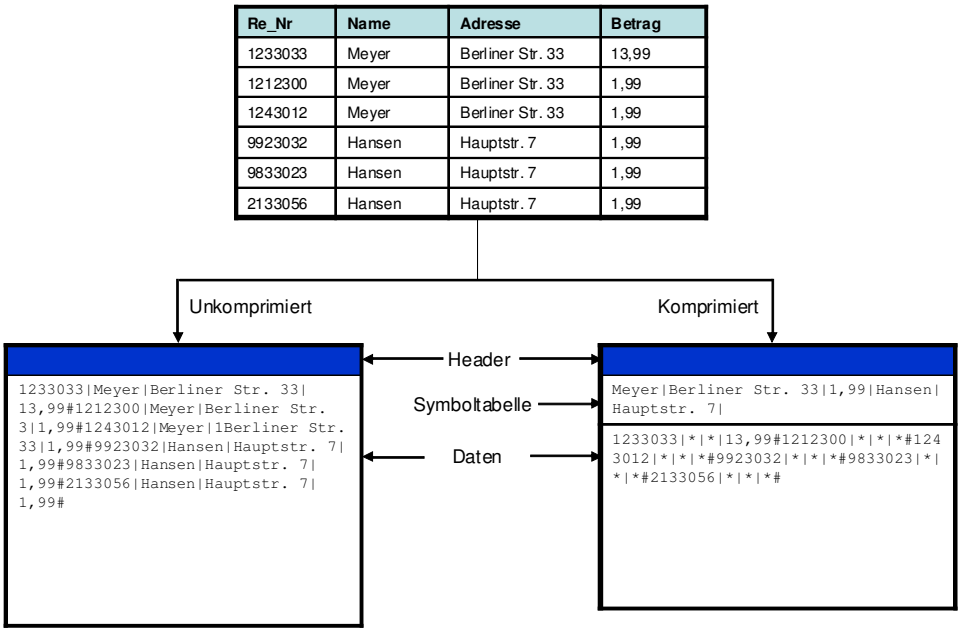

Abb. 3: Komprimierte und unkomprimierte Speicherung

Abbildung 3 zeigt das Kompressionsprinzip: Im oberen Teil befinden sich Beispieldaten einer relationalen Tabelle. Links ist die standardmäßige, unkomprimierte physische Organisation dieser Datensätze in einem Block zu sehen. Hierbei besteht der Block aus einem Header mit Metadaten und die Datensätze werden fortlaufend zeilenweise abgespeichert. In der rechten Abbildungshälfte ist die komprimierte Speicherung dargestellt. In dieser werden wiederholt auftretende Werte in einer Symboltabelle am Blockanfang hinter dem Header abgelegt. Bei den eigentlichen Datensätzen werden

dann lediglich Verweise auf die Symboltabelle eingetragen, was in der Abbildung als * symbolisiert ist.
Zur Reduzierung des Overhead wird Kompression nicht bei jeder Einfüge- bzw. Aktualisierungsanweisung berücksichtigt. Vielmehr wird folgender Algorithmus angewendet: Zunächst werden Datensätze unkomprimiert abgespeichert. Ist ein bestimmter Schwellwert erreicht, werden alle Datensätze im Block komprimiert. Anschließend neu eingefügte Datensätze werden wiederum unkomprimiert abgespeichert bis der Schwellwert erneut erreicht ist. Dieser Prozess wiederholt sich, bis der Datenbank-Server feststellt, dass durch weitere Kompression dieses Blocks kein weiterer Vorteil entsteht. Diese Vorgehensweise ist in Abbildung 4 dargestellt.

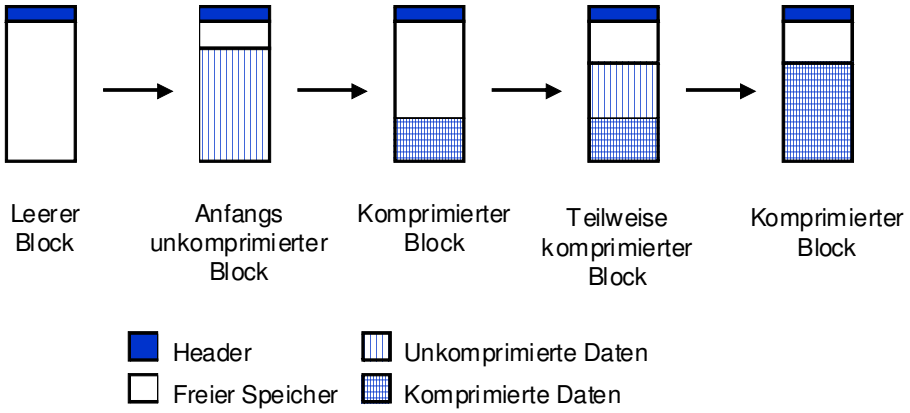

Abb. 4: Algorithmus zur Kompression innerhalb eines Blocks

Aktiviert wird die Kompression durch die `COMPRESS FOR OLTP` oder die `COMPRESS FOR ALL OPERATIONS`-Klausel in der `CREATE TABLE`-Anweisung, z.B.:

```
CREATE TABLE myTable(
   …)
COMPRESS FOR ALL OPERATIONS;
```

Die beiden Klauseln können auch später im Rahmen einer `ALTER TABLE`-Anweisung verwendet werden. Dann wird die Kompression aber nur auf ab diesem Zeitpunkt eingefügte Daten angewendet, bereits vorhandene Daten bleiben unkomprimiert.

Ebenso kann die Kompressionsklausel auf Tablespace-Ebene angewendet werden, dann werden alle in diesem Tablespace angelegten Tabellen komprimiert.

3 Untersuchungen

Generelle Aussagen zu erreichbaren Kompressionsfaktoren sind schwer zu treffen, da diese von vielen Parametern, wie z.B. Häufigkeit unterschiedlicher Werte, Reihenfolge der Dateneingabe, etc. abhängen. Daher werden in diesem Abschnitt ein paar praktische Untersuchungen vorgestellt, um zu zeigen, welche Kompressionsfaktoren prinzipiell erzielt werden können, wovon diese abhängen und ob es Möglichkeiten gibt, den Kompressionsfaktor zu beeinflussen.

3.1 Ähnliche, aber nicht gleiche Daten

Als Beispiel für eine Tabelle mit sehr ähnlichen (aber nicht gleichen) Daten wurden fortlaufend Werte nach dem Muster 13-stelliger ISBN-Werte (xxx-x-xxx-xxxxx-x, wobei jedes x eine Ziffer zwischen 0 und 9 ist) generiert. Da Oracle Advanced Option lediglich auf Gleichheit der Attributwerte prüft, findet hier keine Kompression statt.

In einem zweiten Durchgang wurden die einzelnen Komponenten eines ISBN-Wertes in separaten Feldern abgespeichert, was zu einem Kompressionsfaktor von ca. 1,8 führte.
Dieses Beispiel zeigt, dass durch vertretbare Änderungen am DB-Entwurf der Kompressionsfaktor gesteigert werden kann.

3.2 Transaktionsdaten

Transaktionsdaten sind dadurch gekennzeichnet, dass jeder Datensatz eine Datums-/Zeitangabe enthält, die Daten (relativ gut) sortiert nach diesem Attribut in die Datenbank eingefügt werden und die Datensätze relativ kurz sind. Als Beispiel soll hier ein Datensatz wie im Schema im linken Teil der Abbildung 5 dienen, wie er zum Beispiel von Ticketautomaten erzeugt werden kann.

Tabelle Event	
Attribut	**Typ**
ID	**NUMBER(10)**
Event_Time	DATE
Terminal_ID	NUMBER(10)
Value	NUMBER(10,2)

Tabelle Event_Separiert	
Attribut	**Typ**
ID	**NUMBER(10)**
Event_Year	NUMBER(4)
Event_Month	NUMBER(12)
Event_Day	NUMBER(2)
Event_Hour	NUMBER(2)
Event_Minute	NUMBER(2)
Event_Second	NUMBER(2)
Terminal_ID	NUMBER(10)
Value	NUMBER(10,2)

Abb. 5: Transaktionale Daten

Diese transaktionalen Daten werden gar nicht komprimiert, Ursache hierfür ist die Form der Speicherung des Datentyps `DATE` in Oracle als Kombination aus Datums- und Zeitangabe und der Tatsache, dass für die Kompression nur exakt gleiche Werte in Frage kommen. Auch eine Separierung der Felder wie in Abbildung 5 rechts dargestellt führt zu keiner Verbesserung, da die einzelnen Werte zu klein sind. Im Gegensatz zum Beispiel in Abschnitt 3.1 hat die Schemaänderung hier keinen Vorteil für die Kompression ergeben.

3.3 Fakttabelle

Fakttabellen in einem Data Warehouse bieten sich für Kompression besonders an, denn sie sind zum einen „groß“, so dass sich Kompression lohnt und zum anderen besitzen sie häufig eine Reihe redundanter Attribute. Eine Fakttabelle besitzt typischerweise einen zusammengesetzten Primärschlüssel und eine Reihe Attributen der Typen Integer, Real und Aufzählungstypen mit kleinem Wertebereich.

Tabelle F1		
Attribut	**Typ**	**Wertebereich**
Dim1_ID	**NUMBER(10)**	**Zusammengesetzter Primärschlüssel**
Dim2_ID	**NUMBER(10)**	
Dim3_ID	**NUMBER(10)**	
IntAttribut_1	NUMBER(10)	0-2
IntAttribut_2	NUMBER(10)	1-3
IntAttribut_3	NUMBER(10)	1-10
EnumAttribut_1	VARCHAR2	red, yellow, green
EnumAttribut_2	VARCHAR2	true, false
EnumAttribut_3	VARCHAR2	cheese, cloth, toys, sale, foto, cosmetics, furniture, fruit, meat, frozen
RealAttribut_1	NUMBER(10,2)	0,0 – 2,0
RealAttribut_2	NUMBER(10,2)	0,0 – 2,0
RealAttribut_3	NUMBER(10,2)	0,0 – 2,0
RealAttribut_4	NUMBER(10,2)	1,0 – 10,0
RealAttribut_5	NUMBER(10,2)	1,0 – 10,0

Datensätze	Blöcke Unkomprimiert	Blöcke Komprimiert	Faktor
100000	249	123	2,0
200000	469	217	2,2
500000	1131	532	2,1
1000000	2268	1129	2,0

Abb. 6: Fakttabelle F1: Struktur und Kompressionsfaktor

Abbildung 6 zeigt Fakttabelle F1, bestehend aus drei Integer-, fünf Real-Attributen und drei Attributen mit Aufzählungsdatentyp. Tabelle F1 weist mit 2,0 einen relativ niedrigen Kompressionsfaktor auf. Dies liegt insbesondere an den Attributen vom Typ Real, da hier exakt gleiche Werte natürlich selten auftreten.
Ebenso können die Attribute des zusammengesetzten Primärschlüssels nicht komprimiert werden. Die Variante hier einen zusätzlichen künstlichen Primärschlüssel einzufügen und über die drei Attribute mit der Endung „_ID“ ein `UNIQUE`-Constraint zu definieren, führten jedoch zu ähnlichen Kompressionsfaktoren, ebenso die Variante mit Weglassen des `UNIQUE`-Constraint. Ebenso haben hier verschiedene Blockgrößen keinen Einfluss auf den Kompressionsfaktor gehabt.

Tabelle F2 bis F4				
Attribut	**Typ**	**Werte in F2**	**Werte in F3**	**Werte in F4**
Dim1_ID	**NUMBER(10)**	**Zusammengesetzter Primärschlüssel**		
Dim2_ID	**NUMBER(10)**			
Dim3_ID	**NUMBER(10)**			
IntAttribut_1	NUMBER(10)	1-10	1001-1010	100001-100010
IntAttribut_2	NUMBER(10)	1-10	1001-1010	100001-100010
IntAttribut_3	NUMBER(10)	1-10	1001-1010	100001-100010
IntAttribut_4	NUMBER(10)	1-10	1001-1010	100001-100010
IntAttribut_5	NUMBER(10)	1-10	1001-1010	100001-100010
IntAttribut_6	NUMBER(10)	1-10	1001-1010	100001-100010
IntAttribut_7	NUMBER(10)	1-10	1001-1010	100001-100010
IntAttribut_8	NUMBER(10)	1-10	1001-1010	100001-100010
IntAttribut_9	NUMBER(10)	1-10	1001-1010	100001-100010
IntAttribut_10	NUMBER(10)	1-10	1001-1010	100001-100010

Datensätze	Tabelle F2			Tabelle F3			Tabelle F4		
	Blöcke Unkomprimiert	Blöcke Komprimiert	Faktor	Blöcke Unkomprimiert	Blöcke Komprimiert	Faktor	Blöcke Unkomprimiert	Blöcke Komprimiert	Faktor
100000	172	91	1,9	217	91	2,4	280	91	3,1
200000	343	154	2,2	406	154	2,6	532	154	3,5
500000	879	375	2,3	1036	375	2,8	1320	375	3,5
1000000	1729	753	2,3	2013	753	2,7	2778	753	3,7

Abb. 7: Fakttabelleen F2 bis F4: Struktur und Kompressionsfaktor

Höhere Kompressionsfaktoren werden bei den in Abbildung 7 dargestellten Fakttabellen F2 bis F4 erreicht, in denen jeweils 10 Faktattribute vom Typ Integer vorliegen. Der Kompressionsfaktor erhöht sich hier durch die Länge der Werte. Während in allen drei Fällen bei jedem Attribut 10 verschiedene Werte möglich sind, liegen diese bei Tabelle F2 zwischen 1 und 10 (Kompressionsfaktor 2,3), bei Tabelle F3 liegen die Werte zwischen 1001 und 1010 (Kompressionsfaktor 2,7). In Tabelle F4 schließlich liegen die möglichen Werte für jedes Faktattribut zwischen 100001 und 100010, was zu einem Kompressionsfaktor von 3,5 führt. Der steigende Kompressionsfaktor in den Tabellen F2 bis F4 erklärt sich dadurch, dass ein größerer Einzelwert mehr Platz einnimmt, dies aber bei der Kompression keine Rolle spielt (abgesehen von minimal mehr benötigtem Platz in der Symboltabelle).

3.4 Vergleich zur Zip-Kompression

Als letztes Beispiel soll der von Oracle Advanced Compression erzielte Kompressionsfaktor mit dem des Zip-Standardalgorithmus verglichen werden. Dazu wurden jeweils 100000 Zeichenketten der Länge m generiert, wobei in einer solchen Kollektion n verschiedene Werte auftreten. Dies wurde für 200000, 500000 und 1000000 Datensätze wiederholt. Das Resultat ist in Abbildung 8 dargestellt.

n	m	Komp.faktor Oracle	Komp.faktor Zip
2	2	1,0	16,3
2	10	1,4	48,0
2	50	4,4	144,0
5	2	1,0	7,2
5	10	1,3	21,3
5	50	4,6	71,7
10	2	1,0	5,3
10	10	1,4	14,9
10	50	4,0	49,3
50	2	1,0	3,1
50	10	1,3	9,1
50	50	2,6	26,0
100	2	1,0	2,7
100	10	1,0	7,3
100	50	1,5	22,1

Abb. 8: Vergleich mit Zip-Kompression

Ein Vergleich der dritten und vierten Spalte zeigt den deutlich höheren Kompressionsfaktor des Zip-Algorithmus. Weiterhin ist aus den gemessenen Werten zu ersehen, dass Oracle Advanced Compression erst bei relativ langen Zeichenketten (großer Wert m) einen guten Kompressionsfaktor aufweist. Bei kurzen Zeichenketten (Parameter m<=10) ist der Kompressionsfaktor kleiner 1,5, unabhängig von der Anzahl verschiedener Werte.
Weiterhin ist zu beobachten, dass mit der Anzahl unterschiedlicher Werte (wachsender Parameter n) der Kompressionsfaktor sinkt. Eine generelle Aussage ist auch hier schwer möglich, ab welchem Wert sich eine Kompression nicht mehr lohnt, weil der Kompressionsfaktor zu gering ist.

3.5 Andere Untersuchungen

Die bei den Untersuchungen in den vorhergehenden Abschnitten erzielten Ergebnisse werden durch andere Untersuchungen bestätigt:

- In [Orac09] wird für nicht näher spezifizierte „normale OLTP-Daten“ ein Kompressionsfaktor von 2 bis 3 genannt.
- [Nand08] erzielt für eine synthetisch erzeugte Stammdatentabelle einen Kompressionsfaktor von 2,75. Hierbei ist anzumerken, dass die Attributwerte innerhalb einer Spalte eine relativ hohe Wiederholungsrate besitzen.
- [PoPo03] erzielen für reale Daten einen Faktor von 3 bis 4 und geben für die Lineitem-Fakttabelle einen Faktor von 1,6 an.

4 Zusammenfassung und Ausblick

In diesem Beitrag wurde zunächst eine Motivation für Kompression in Datenbanken gegeben, dann das in Oracle 11g R2 realisierte Advanced Compression Konzept beschrieben und bei der Kompression erreichte Resultate aus der Literatur und eigenen Untersuchungen dargestellt.
Dabei kann festgehalten werden, dass (in Abhängigkeit der Daten) Kompressionsfaktoren von 1,0 (keine Kompression) bis etwa 4,0 erreicht werden können.

Als generelle Vorteile der Datenbankkompression können festgehalten werden:

- Das reduzierte Volumen führt zur Einsparung von Plattenplatz. Da (hochwertige) Festplatten in modernen Serversystem kostenintensive Komponenten sind, lassen sich somit die Hardwarekosten senken.
- Weil der Energieverbrauch in Rechenzentrum proportional zur Anzahl der eingesetzten Festplatten und der Anzahl der CPUs ist, kann die geringere Anzahl an benötigten Festplatten den Energiebedarf reduzieren.
- Kleinere Datenbanken führen zu kleineren Backups und als Konsequenz können sowohl die Sicherungs- als auch Wiederherstellungszeiten reduziert werden.
- Wesentlicher Faktor für die Antwortzeit von Anfragen ist die Anzahl der von der Festplatte zu lesenden Blöcke. Da komprimierte Daten weniger Platz benötigen, müssen bei bestimmten Anfragen (z.B. Full Table Scans) weniger Blöcke gelesen werden und es ergeben sich schnellere Antwortzeiten.
- Der Datenbankpuffer ist stets eine kritische Ressource, um welche parallele Anfragen konkurrieren. Komprimierte Daten sparen Platz im Puffer (es befinden sich bei gleicher Blockzahl mehr Datensätze im Puffer), dieser kann für andere Anfragen genutzt werden und somit wird die Trefferrate erhöht.

Diesen Vorteilen steht ein Overhead in Form langsamerer Einfüge- und Aktualisierungsoperationen gegenüber. Dieser ist in Oracle 11g zwar messbar, fällt aber nicht wesentlich ins Gewicht.
Zusammenfassend kann damit zu Oracle Advanced Compression gesagt werden, dass nicht die höchst möglichen Kompressionsfaktoren erzielt werden, sondern vielmehr auf eine Reduzierung des Overhead in Form zusätzlicher CPU-Last und daraus folgenden langsameren Einfüge- und Aktualisierungsoperationen Wert gelegt wurde.
Höhere Kompressionsfaktoren ließen sich erzielen, wenn nicht nur pro Attribut gleiche Werte komprimiert würden, sondern auch Teilworte (siehe Beispiel mit ISBN-Nummern). Eine weitere Verbesserung wäre durch eine tabellenbezogene statt einer blockbezogenen Kompression zu erwarten. Während diese beiden Ansätze den Kompressionsfaktor „etwas“ erhöhen, würde die Möglichkeit einer spaltenorientierten Speicherung (wie z.B. in Sybase IQ oder Oracle Exadata realisiert) zu einer wesentlichen Verbesserung des Kompressionsfaktors beitragen, was durch die Homogenität der Daten in einer Spalte begründet ist.

Kontaktadresse:

Name
Duale Hochschule BW Campus Horb
Florianstr. 15
D-72160 Horb

Telefon: +49 (0) 7451-521 146
Fax: +49 (0) 7451-521 101
E-Mail o.herden@hb.dhbw-stuttgart.de
Internet: www.dhbw-stuttgart.de/horb

Referenzen

[CGH+08]. Carr, B., Garmany, J., Hartmann, L., Jain, V. J., Karam, S.: Oracle 11g New Features: Get Started Fast with Oracle 11g Enhancements. Rampant Techpress, Kittrell(North Carolina) (2008)

[Mann09] Mann, M.: Kompression, Ausführungspläne und Bind-Peeking. DOAG News, S.43-44, Q2/2009.

[Nand08] Nanda, A.: Compress to Impress – Use Oracle Advanced Compression to Conserve Resources and Improve Performance. Oracle Magazine, S.55-58, Juli/August 2008.

[Orac09] Oracle: Advanced Compression with Oracle Database 11g Release 2. Oracle white paper, 2009. Online verfügbar unter http://www.oracle.com/technology/products/database/oracle11g/pdf/

[PoPo03] Poess, M.,Potapov, D.: Data Compression in Oracle. VLDB Proceedings 2003, S. 937-947. Online verfügbar unter http://www.vldb.org/conf/2003/papers/S28P01.pdf

[Salo06] Salomon, D. Data Compression: The Complete Reference. Springer, Berlin (2006)

[Sayo05] Sayood, K. Introduction to Data Compression. Morgan Kaufmann, San Francisco (2005)

[ZiLe77] Ziv, J., Lempel, A.: A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory 23(3): 337-343 (1977)