Oracle Advanced Compression Option

Martin Bracher
OPITZ CONSULTING Schweiz GmbH
Thalwil, Zürich

Schlüsselworte:

Advanced Compression, Backup, RMAN, Data Guard, Table Compression, Secure Files, LOB, Hybrid Columnar Compression, Exadata

Einleitung

Unsere Datenbestände wachsen unaufhaltsam. Wir speichern immer mehr wichtige (und auch unwichtige) Daten. Wenn wir mal annehmen, wir müssen oder wollen diese Daten speichern, dann bieten sich uns zwei Lösungswege. Wir benutzen mehr und grössere Disks, oder wir komprimieren die Daten.
Mit der Advanced Compression Option bietet uns Oracle verschiedene Möglichkeiten der Datenkomprimierung. In diesem Vortrag werden die verschiedenen Varianten der Komprimierung erörtert:

- Komprimierung von Tabellen
- Komprimierung unstrukturierter Daten (Securefiles)
- Backup-Komprimierung
- DataGuard Netzwerkkomprimierung

Ebenso werden deren Vor- sowie auch deren Nachteile betrachtet.

Wie der Namensbestandteil „Option“ schon vermuten lässt, handelt es sich bei der Advanced Compression Option um eine kostenpflichtige Option und liegt im Bereich von etwa 24% des Preises der Enterprise Edition.

Komprimierung von Tabellen

Die Komprimierung von Tabellen (genauer gesagt, die Komprimierung der Records in einem Tabellenblock) ist grundsätzlich nichts Neues, sie wurde mit 9i eingeführt und ist Bestandteil der Enterprise Edition:

```
create table tab1
(field1 number, field2 varchar2(30))
compress;
```

Diese Komprimierung war aber starken Einschränkungen unterworfen. Die Blöcke wurden nur komprimiert, wenn die Tabelle mit Direct-Load Operationen gefüllt wurde.

```
create table tab1 as select …
insert /*+ append */ into tab1 …
```

Bei gewöhnlichen Inserts / Updates wurden die Daten hingegen nicht komprimiert. Diese Möglichkeit wurde erst mit 11g eingeführt und wird als OLTP-Compression bezeichnet.

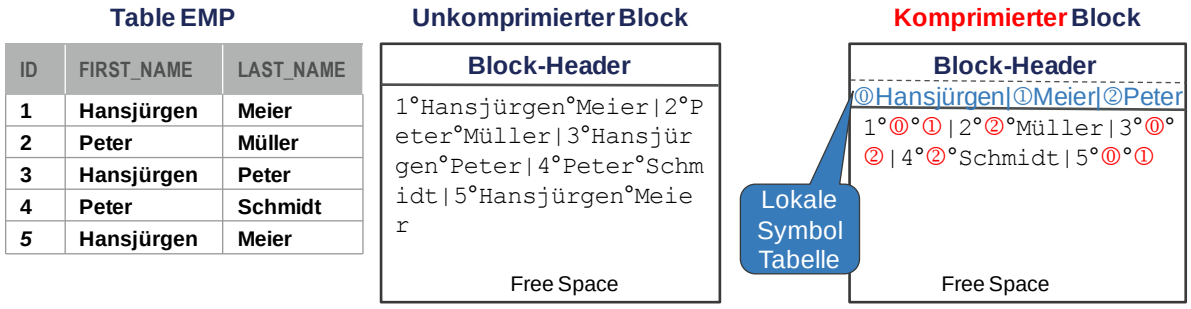
```
create table tab1
(field1 number, field2 varchar2(30))
compress for all operations;
```

Die Syntax hat sich mit den Oracle-Versionen verändert:

Version	Bulk (direct) Load	OLTP
9.2	`compress`	-
11.1	`compress [for direct_load operations]`	`comress for all operations`
11.2	`compress [basic]`	`compress for oltp`

Art der Komprimierung

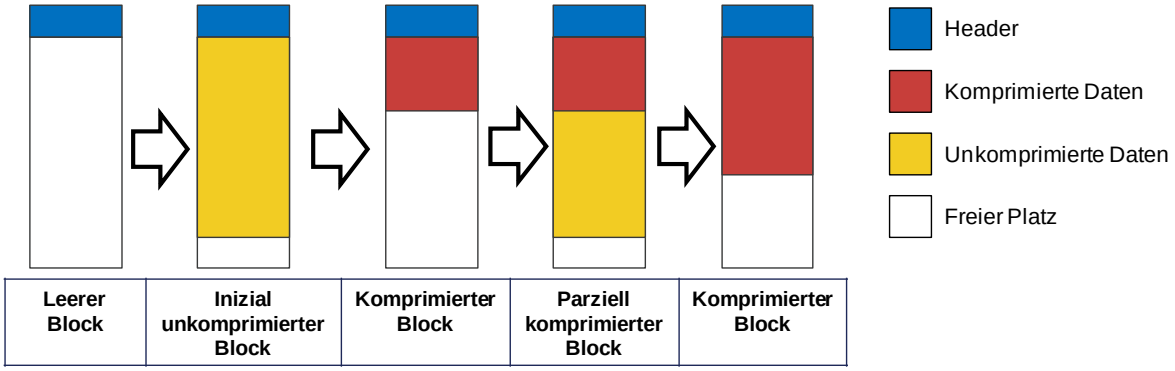
Die Komprimierung erfolgt pro Oracle Block. Innerhalb des Blocks werden doppelte Werte gesucht, und wenn gefunden, werden diese einmalig in eine sogenannte Symboltabelle eingefügt. Der Feldinhalt wird danach durch einen Verweis auf die Symboltabelle ersetzt.
Diese Art der Komprimierung erfolgt nicht pro Feld, sondern über alle Felder. Sollte jemand „Peter“ als Nachname haben, und jemand anderes als Vorname, dann werden beide durch den Verweis auf denselben Symboltabellen-Eintrag komprimiert.



Da die Komprimierung pro Block erfolgt, sind aus Sicht der Komprimierung grosse Blöcke von Vorteil. Denn je mehr gleiche Felder in einem Block vorkommen, desto grösser ist die Komprimierungsrate. Diese können wir uns durch eine Funktion des Compression Advisors (Download) ausgeben lassen:

```
exec dbms_comp_advisor.getratio (ownername=>'TEST', tabname=>'T1BASIC', -
                compress_type=>'DIRECT_LOAD', sampling_percent=>20);
```

Um den Overhead der Komprimierung zu reduzieren, werden die Blöcke nicht bei jedem Insert oder Update wieder neu komprimiert. Solange die Daten auch unkomprimiert in einem Block Platz haben, spielt es ja für die Leseperformance keine Rolle, wenn die Daten teilweise unkomprimiert sind. Erst nach einigen Änderungen und einem bestimmen Füllstand wird Oracle den Block wieder neu komprimieren. Nur diese Transaktion, die die Komprimierung ausgelöst hat, wird etwas langsamer; die anderen haben die übliche Performance vom Schreiben nicht komprimierter Blöcke.



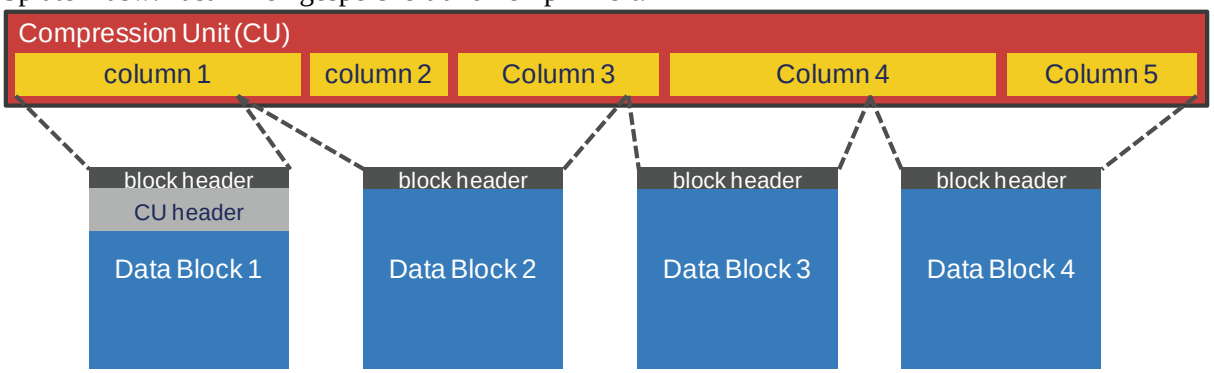
Vor- und Nachteile:
Der geringere Platzverbrauch ist sicher ein gewichtiger Vorteil. Die geringere Anzahl Blöcke kann sich aber auch positiv auf die Geschwindigkeit auswirken:

- Bei einem Full-Table Scan müssen weniger Blöcke gelesen werden.
- Da pro Block mehr Rows gespeichert sind, ist auch die Wahrscheinlichkeit grösser, dass sich der benötigte Datensatz bereits in der SGA befindet und dadurch ein Zugriff auf Disk vermieden werden kann.

Nachteilig kann sich jedoch die zusätzliche CPU-Belastung für die (De-)Komprimierung auswirken. Auch Row-Migration (nach Update hat die Row nicht mehr im ursprünglichen Block Platz) kann zu einem Problem werden.
Falls die Komprimierung nicht so funktioniert, wie man sich das vorstellt, dann kann das möglicherweise auch an einigen Einschränkungen liegen. Beispielsweise lässt sich Compression nicht verwenden, wenn die Tabelle mehr als 255 Felder hat, oder wenn sie im SYSTEM Tablespace liegt.
Im Vortrag werden wir anschauen, wie sich die Komprimierung auf die Datenmenge und den CPU- bzw. Zeitverbrauch auswirken kann.

Hybrid Columnar Compression

Kürzlich hatte ich die Gelegenheit, einige Messungen auf einer einer Oracle Exadata Database Machine machen zu können. Wirklich interessant wird die Komprimierung nämlich da drauf. In diesem Fall steht eine weitere Komprimierungsmöglichkeit zur Verfügung, die Hybrid Columnar Compression. Im Gegensatz zur oben beschriebenen Komprimierung erfolgt hier die Komprimierung nicht mehr pro Block über alle Felder. Die Datensätze werden in sogenannten Compression Units, einer Menge von Blöcken gespeichert. Dabei werden zuerst alle Felder der Spalte 1, dann alle der Splate 2 usw. zusammen gespeichert und komprimiert.



Dies ermöglicht in der Regel eine viel effizientere Komprimierung, da die Wahrscheinlichkeit für Mehrfach-Werte innerhalb einer Spalte grösser ist als innerhalb eines Datensatzes. Diese Speicherung

nach Spalten hat auch noch ganz andere Vorteile: Wenn wir nur ein Feld selektieren, müssen auch nur die Blöcke gelesen werden, die die entsprechende Spalte enthält.
Bei der Komprimierung erhalten wir 4 Varianten zur Auswahl, die sich durch den CPU-Verbrauch und den Platzbedarf unterscheiden.

	Komprimierungsart	Zweck	CPU-Verbrauch	Komprimierung
1	comrpess for query low	DWH	niedrig	gut
2	compress for query high	DWH	mittel	besser als 1
3	compress for archive low	Langzeitarchivierung	mittel	minimal besser als 2
4	compress for archive high	Langzeitarchivierung	hoch	am Stärksten

External Tables

Bei External Tables hilft uns die Oracle Komprimierung nichts, wir können aber die Komprimierung auf Betriebssystem-Ebene und das 11.2 Preprocessing Feature verwenden. Die Daten der external Table liegt dabei komprimiert auf dem Filesystem und wird erst beim Lesen durch das Pre-processing Programm dekomprimiert. Dieses Programm kann beispielsweise das zcat Utility vom Betriebssystem sein.

```
CREATE TABLE exttab (ID number, NAMES varchar2(30))
ORGANIZATION external (TYPE oracle_loader
 DEFAULT DIRECTORY data_pump_dir
 ACCESS PARAMETERS  (RECORDS DELIMITED BY NEWLINE
  PREPROCESSOR data_pump_dir:'zcat'
  FIELDS TERMINATED BY "|" (ID, NAMES)) location ('exttab.gz'));
```

Secure Files - Komprimierung unstrukturierter Daten

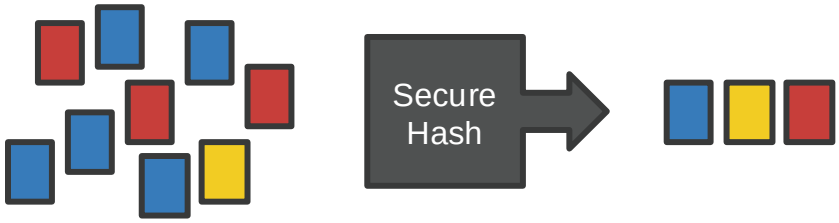
So ein Titel, wo die meisten wohl nicht recht wissen, was damit eigentlich gemeint ist. Unter unstrukturierten Daten sind beliebige (Binär)daten zu verstehen, beispielsweise ein PDF-Dokument oder ein Bild im jpeg-Format. Solche Dateien werden in einem Oracle LOB Feld abgespeichert. Die LOB's können wie bisher gespeichert werden, was heute als „BASICFILE“ bezeichnet wird, oder mit der neuen „SECUREFILE“ Technologie. Für die Benutzung der LOB's sind die Interfaces transparent. Und „secure“ heisst es wohl, weil der Inhalt verschlüsselt werden kann (Advanced Security Option).
„SECUREFILE“ ist aber auch die Technologie, die uns erlaubt, den Inhalt zu komprimieren.

```
create table tab1 (field1 BLOB)
LOB(field1) store as securefile (tablespace L1 compress [low|medium|high])
```

Securefiles müssen in einem Automatic Segment Space Management Tablespace liegen.

Neben dieser klassischen Komprimierungsmöglichkeit von Binärdateien gibt es auch noch eine Komprimierung, die vom Prinzip her ähnlich wie bei der Table Compression funktioniert. Statt gleiche Felder nur 1x zu speichern, werden hier gleiche Dateien jeweils nur 1x gespeichert. Man spricht von Deduplication.

```
alter table tab1 modify LOB(field1) (deduplicate);
```



Im Vortrag werden wir uns anschauen, wie gut die Komprimierung in Abhängigkeit verschiedener Datei-Typen und -Grössen ist, aber auch im Vergleich zu einer Komprimierung auf Betriebssystem-Ebene mit beispielsweise gzip. Wer seine Vergleiche mit den Byte-Werten aus dba_segments macht, wird zuerst mal staunen, denn die Messwerte sind nicht reproduzierbar... Wir müssen auch andere Messmethoden berücksichtigen. Zur Veranschaulichung ein Beispiel eines Loads mit sqlldr:

Komprimierung	Ladezeit	MB aus dba_segments	effektive Grösse
LOW	11-15 sec.	129/123/130	109.1
MEDIUM	13-16.5 sec.	113/113/114	98.6
HIGH	17-18.5 sec.	106/113/113	94.3
ohne	16-16.5 sec	192/184/184	162.4

Auf der Hardware in diesem Beispiel ist Low- und Medium-Komprimierung schneller als die unkomprimierte Speicherung, weil weniger Daten geschrieben werden müssen. Bei der High-Komprimierung ist jedoch der Zeitbedarf für die Komprimierung grösser als der Zeitgewinn durch das Speichern einer kleineren Datenmenge.

Aufgrund verschiedener Messresultate werde ich im Vortrag versuchen, eine „Best Practise“ für die Komprimierung weiterzugeben: Unter welchen Voraussetzungen soll ich überhaupt komprimieren? Wann ist Oracle Compression die Wahl, oder wann erfolgt die Komprimierung besser auf Client-Seite.

Backup-Komprimierung

RMAN

Die RMAN Backup-Komprimierung ist nichts Neues und auch ohne die Advanced Compression Option (AC) nutzbar. Die Backupsets werden mit dem BZIP2 Algorithmus sehr stark komprimiert, jedoch verbunden mit starker CPU-Belastung.

```
rman> backup as compressed backupset database ...
```

Mit der AC erhalten wir nun einen weiteren Kompressionsalgorithmus, den ZLIB-Algorithmus, der etwas weniger stark komprimiert, dafür aber auch weniger CPU-Ressourcen benötigt.

```
rman> configure compression algorithm to 'bzip2|zlib';          #11.1
rman> configure compression algorithm 'basic|low|medium|high'; #11.2
rman> backup as compressed backupset database ...
```

Die Kompressionsraten sind durchaus beträchtlich. Zur Illustration: Auf einer kleinen Datenbank sieht die Komprimierung wie folgt aus.

no compression	basic	low	medium	high
1601 M / 100%	553 M / 34.5%	657 M / 41.0%	583 M / 36.4%	508 M / 31.7%

Abhängig vom Dateninhalt können die Kompressionsraten auch besser (viel Text), oder schlechter (Jpeg Fotos in der Datenbank) sein.

Im Vortrag werden wir uns auch damit beschäftigen: Lohnt sich die AC Option überhaupt, wenn wir sie nur fürs Backup benötigen? Brauchen wir sie überhaupt oder ist das Geld nicht besser in zusätzliche CPU-Lizenzen (BASIC-Komprimierung) oder zusätzlichen Diskplatz (unkomprimiert) investiert? Macht Backup-Komprimierung bei Tape-Backup Sinn?

Export

Das alte „exp“ Utility kannte keine Komprimierung, aber wir konnten (jedenfalls auf geeigneten Betriebssystemen) die Komprimierung ohne physische Zwischenspeicherung selbst implementieren

```
mkfifo exp.dmp && gzip exp.dmp&
exp file=exp.dmp …
```

Die neue Datapump-Funktionalität „expdp“ ist jedoch nicht mehr fähig, in eine Pipe zu schreiben, weil die Daten nicht mehr in einem sequenziellen Stream geschrieben werden. Die Konvertierung muss somit zwingend von Oracle gemacht werden.

Mit Version 10.2 gab es zwar eine COMPRESSION Option. Diese war aber nur in der Lage, die Metadaten, nicht jedoch die meist viel grösseren Tabellendaten zu komprimieren.

Mit Oracle 11 hat sich das nun gebessert und wir können auch den Tabelleninhalt komprimieren, aber leider erfordert auch dies die kostenpflichtige AC Option.

Data Guard Netzwerk-Komprimierung

Die Übertragung der Änderungen von der Primary- zur Standby-Database kann synchron oder asynchron in Form eines kontinuierlichen Redo-Streams, oder durch Transfer der Archivelogs erfolgen. Falls der Standby Änderungen fehlen (beispielsweise wegen Netzwerkunterbruch), kann sie diese fehlenden Archivelogs nochmals anfordern.

Alle diese Transfers erfolgten bis Version 10 unkomprimiert. Mit 11g Release 1 wurde zwar offiziell die Komprimierung für den Transfer eingeführt, aber die Implementierung war nicht wirklich so, wie man sich das wünschte. Es wurde nämlich nur beim Anfordern fehlender Archivelogs komprimiert (und dies ist hoffentlich nur ein Ausnahmefall); Wieso aber nicht mal ein regulärer Archivelog-Transfer komprimiert erfolgte, war mir unverständlich. Erst mit 11.2 kann nun auch der normale Transfer, sowohl Redo als auch Archive, komprimiert werden, und somit kann die erforderliche AC Option nun möglicherweise auch sinnvoll genutzt werden.

Die Komprimierung ist ein Database Property, welches in der log_archive_dest_n enthalten ist.

```
DGMGRL> edit database 'DB1120_RZ1' set property RedoCompression='ENABLE';
DGMGRL> show database 'DB1120_RZ1' RedoCompression
```

Im Vortrag werden wir die zu erwartende Komprimierung anschauen und uns überlegen, wann der Einsatz sinnvoll ist.

Fazit

Ob der Einsatz der Advanced Compression Option Sinn macht, kann nicht generell gesagt werden. Das hängt stark von der eigenen Umgebung ab. Es gibt Daten, die lassen sich hervorragend komprimieren, andere sind kaum komprimierbar (jpeg-Fotos). Komprimierung bewirkt eine Reduktion des Disk-I/O, jedoch wird für die (De-)Komprimierung mehr CPU benötigt. Je nach Art der Daten und Performance des Storage und der CPU kann durch Komprimierung eine Geschwindigkeitsverbesserung, aber auch eine Verschlechterung resultieren. Um eigene Messungen auf der eigenen Umgebung wird man nicht herumkommen.

Wenn die Komprimierung technisch Vorteile bringt, müssen wir uns auch überlegen, ob diese kostenpflichtige Option wirtschaftlich ist. Möglicherweise ist etwas mehr Storage günstiger, oder der Performance-Gewinn bietet keinen direkten Vorteil. Beispielsweise der Batch-Job in der Nacht ist 10 Minuten früher fertig und niemanden interessierts.

Kontaktadresse:

Martin Bracher
OPITZ CONSULTING Schweiz GmbH

Seestrasse 97
CH-8800 Thalwil

Telefon: +41 44 721 11 22
E-Mail martin . bracher@opitz-consulting.com
Internet: www.opitz-consulting.com