Komprimierung in der Datenbank - aktueller Stand, Neuigkeiten, Einsatzmöglichkeiten

Ulrike Schwinn
Oracle Deutschland B.V. & Co.KG
München

Schlüsselworte

Basic Compression, OLTP Compression, HCC, LOB Compression, Compression Advisor, RMAN, Data Pump, Netzwerk

Einleitung
Schon seit vielen Jahren ist die Komprimierung von Daten ein wichtiger Bestandteil der Oracle Datenbank und wird beständig weiterentwickelt. Dies zeigte sich besonders auch im Datenbankrelease 11*g* mit der Einführung von neuen Techniken im Zusammenhang mit der neuen Option Advanced Compression. Die Komprimierung ist nun beispielsweise unabhängig vom Anwendungs– Workload und zusätzlich um die Bereiche unstrukturierte Daten, Backup Daten und Netzwerk (im Data Guard Umfeld) erweitert worden.

In Oracle Database 12*c* sind sogar Eigenschaften zur Verbesserung des Storage Management ergänzt worden. Im Wesentlichen handelt es sich dabei um zwei neue Features – die **Heat Map** und die **Automatische Daten Optimierung** (englisch **A**utomatic **D**ata **O**ptimization). Die Heat Map „trackt" Veränderungen und Abfragen auf Zeilen und Segment Ebene und gibt einen detaillierten Überblick über den Zugriff auf die Daten. Die Automatische Daten Optimierung verlagert und/oder komprimiert die Daten gemäß Nutzer definierter Regeln (englisch policies) basierend auf den Informationen, die sich aus der Heat Map ergeben. Beide zusammen helfen dabei, **Information Lifecycle Management** Strategien in – und nicht außerhalb – der Datenbank zu implementieren.

Da eine ausführliche Beschreibung aller Funktionen den Rahmen des Artikels sprengen würde, konzentriert sich der folgende Artikel auf eine kurze Definition, einige wichtige Fragestellungen und den aktuellen Stand der Techniken in 12c. Weitere Informationen finden sich im letzten Abschnitt oder im Dojo zum Thema „Komprimierung in der Datenbank".

Tabellen, LOB, Index und Netzwerk Komprimierung
Daten in der Datenbank werden entweder als strukturierte oder als unstrukturierte Daten gespeichert. Um eine effektive Speicherung zu gewährleisten, verwendet die Datenbank dabei unterschiedliche Algorithmen bei der Komprimierung von Daten.

Im Fall von strukturierten Daten handelt es sich um folgende drei Komprimierungsverfahren:

- **Basic (Table) Compression** (auch Direct Load Compression)
- **OLTP Compression** (auch Advanced Row Compression)
- **Hybrid Columnar Compression** (auch Column Store Compression)

Bereits seit Oracle Version 9.2 ist es möglich, relationale Tabellendaten zu komprimieren. Dabei handelt es sich um die sogenannte **Basic Compression** - auch **Direct Load Compression** genannt. Bei der Komprimierung der Daten werden Mehrfacheinträge im Datenblock nur einmal gespeichert. Die sich wiederholenden Werte werden in einer sogenannten „Symbol Table" auf Blockebene gespeichert und durch einen Pointer im Datenteil des Blocks adressiert. Wichtig zu wissen ist, dass

Basic Compression ausschließlich für **Bulk Load Operationen** genutzt werden kann. Dazu gehören folgende Operationen:

- Direct Path Load beim SQL*Loader
- CREATE TABLE AS SELECT
- Paralleler INSERT
- Serieller INSERT mit APPEND Hint und Subquery Klausel (auch Direct Path Insert)

Bitte beachten Sie die Einschränkungen speziell auch bei der Verwendung von Direct Path Inserts. Ein Blick ins SQL Handbuch zeigt, welche Voraussetzungen gelten müssen. Ein typisches Beispiel ist die Verwendung von Constraints. Laut Handbuch gilt nämlich Folgendes: "The target table cannot have any triggers or referential integrity constraints defined on it." Ein Direct Path Insert schaltet also in einen konventionellen Load (INSERT) um, sobald ein referentielles Constraint eingeschaltet ist. Schaltet man das Constraint aus, funktioniert der Direct Path Insert wie zu erwarten ist.

Mit Oracle 11*g* wurde die Basic Compression Methode um ein zweites Verfahren – dem **OLTP Compression** Verfahren – erweitert. Mit diesem Verfahren werden alle DML Operationen – auch konventionelle DML Operationen unterstützt. Komprimiert wird übrigens nicht nach jeder Schreiboperation, sondern in einer Art Batch Mode. Ein neuer Block bleibt so lange unkomprimiert, bis die Daten die PCTFREE Grenze erreicht haben.

Dies bedeutet, dass die Einschränkung auf Bulk Load Operationen entfällt, und Komprimierung ohne Rücksicht auf die Art der Ladevorgänge garantiert werden kann. Diese neue Komprimierungsmethode wird als **OLTP** oder auch als **Advanced Row Compression** (ab 12*c*) bezeichnet.

Worin **unterscheiden** sich die beiden Verfahren? Wie schon aus der Definition bzw. aus der Beschreibung ersichtlich, eignet sich Basic Compression nur für Bulk Load Operationen – also typische Warehouse Applikationen - und nicht für konventionelle DML Operationen. Auch ADD bzw. DROP COLUMN Operationen sind nur bei OLTP Compression zulässig. Tabellen mit mehr als 255 Spalten, Index-Organized Tables, External Tables und Cluster Tables können in 11*g* generell nicht komprimiert werden. In 12*c* wurde allerdings die Spaltengrenze von 255 für Tabellen mit Advanced Row Compression aufgehoben. Im aktuellen Handbuch Oracle Database SQL Language Reference kann man die entsprechenden Einschränkungen nachlesen. Ein weiterer Unterschied ist die Lizenzierung: OLTP Compression erfordert die Lizenzierung der Advanced Compression Option wohingegen Basic Compression schon in der Enterprise Edition enthalten ist.

Gemeinsam hingegen ist der Level der Verwendung. Generell können beide Komprimierungsarten auf **Tablespace-, Tabellen- oder Partitionsebene** mit dem entsprechendem CREATE bzw. ALTER Kommando eingeschaltet werden. Da sich die Syntax Schlüsselwörter bei den Kommandos mehrfach geändert haben, schadet ein Blick in das aktuelle Handbuch nicht. Für 12c gilt folgende Syntax:

OLTP Compression (auch Advanced Row Compression):

```
CREATE TABLE sales_history(…) ROW STORE COMPRESS ADVANCED;
```

Basic Compression:

```
CREATE TABLE sales_history(…) ROW STORE COMPRESS BASIC;
```

Tablespace Compression mit OLTP Compression:

```
CREATE TABLESPACE ... DEFAULT ROW STORE COMPRESS ADVANCED;
```

Die häufig gestellte Frage nach der **Höhe der Komprimierungsrate** lässt sich übrigens nicht allgemein beantworten. Je nach Art der Daten, beispielsweise dem Anteil an redundanten

Informationen, der Datenbank-Blockgröße und der Art des Ladevorgangs können die Komprimierungsraten stark variieren. So ist es mitunter vorteilhaft eine große Datenbank-Blockgröße zu verwenden bzw. mit vorsortierten Daten zu arbeiten, um die Komprimierungsrate zu erhöhen. Um einen Eindruck zu bekommen, wie verschieden oder umgekehrt wie redundant die Daten in den einzelnen Spalten sind, kann die Spalte NUM_DISTINCT aus DBA_TAB_COLUMNS hilfreich sein. Setzt man diese Information ins Verhältnis zu der Anzahl der Zeilen, bekommt man einen recht guten Überblick. Um eine gute Ratio zu erhalten, sollte die zu komprimierende Tabelle nach den Spalten in absteigender Rendundanz umsortiert werden. Beachten sollten Sie dabei auch, dass die Defaultwert für PCTFREE für OLTP und Basic Compression unterschiedlich sind – für OLTP gilt der Default von 10; für Basic Compression 0.

Häufig stellt sich die Frage: wie kann man bestehende Tabellen in ein **komprimierte Format umwandeln**? Möchte man im Nachhinein die Inhalte von existierenden unkomprimierten Tabellen in komprimierte Tabellen umwandeln, kann man die Daten mit folgenden Kommandos in einer Einschritt-Operation umschichten und gleichzeitig komprimieren.

```
ALTER TABLE ... MOVE [TABLE_COMPRESSION_CLAUSE]

ALTER TABLE ... MOVE PARTITION|SUBPARTITION COMPRESS
[TABLE_COMPRESSION_CLAUSE]
```

Allerdings erfordern diese Kommandos bis einschließlich 11*g* Release 2 DML Sperren auf der Tabelle. Falls diese Operationen online (ohne Sperren) erfolgen sollen, muss man mit dem Package DBMS_REDEFINITION arbeiten. Wer partition bzw. subpartitionweise arbeiten kann, kann von der Neuerung in 12*c* profitieren. Die Syntax Erweiterung ONLINE ermöglicht die Ausführung ohne blockierende DML Sperren (TM Lock). Gleichzeitige DML Operationen, die normalerweise in 11*g* durch die entsprechenden Locks blockiert wären, sind nun möglich. Folgendes Beispiel zeigt, wie in 12*c* eine Partition online in das OLTP Komprimierungsformat überführt werden kann.

```
ALTER TABLE sales_big MOVE PARTITION sales_q4_2001 ROW STORE COMPRESS
ADVANCED ONLINE;
```

Nicht verwechseln sollte man die Operationen ALTER TABLE MOVE COMPRESS mit dem Kommando ALTER TABLE ... COMPRESS. Ohne Verwendung des Schlüsselworts MOVE wird nur die sogenannte „Table Property“ geändert, d.h. es werden Speichereinstellungen für die zukünftige Nutzung eingestellt. Nur nachträglich eingefügte Zeilen werden komprimiert abgelegt. Bestehende Zeilen werden allerdings nicht verändert.

Der Einsatz spezieller Storage wie Exadata Storage Systeme, Sun ZFS Storage Appliance (Oracle NAS Storage) oder Axiom Storage (Oracle SAN Storage) ermöglicht die Verwendung von weiteren Komprimierungsalgorithmen, die unter dem Begriff **Hybrid Columnar Compression** zusammengefasst werden. Die Non-Exadata Storage-Systeme (ZFS und Pillar Storage) erfordern dabei eine Datenbankversion ab 11.2.0.3. Oracle verwendet bei HCC eine Kombination aus zeilen- und spaltenbasierter Speicherung und die Verwendung von speziellen Komprimierungsalgorithmen. Die Datensätze werden in logische Compression Einheiten (Logical Compression Unit) aufgeteilt und innerhalb einer Einheit nach Spalten sortiert und danach komprimiert. Die Spaltenwerte einer Gruppe werden dann im selben Datenblock abgespeichert. Auf diese Art und Weise kann eine sehr effiziente Komprimierung erfolgen, und es können hohe Komprimierungsraten – höher als im Fall von BASIC und OLTP Compression - erreicht werden.

Vorgesehen sind diese Verfahren nur für Daten, die nicht häufig verändert werden und sind, wie bei der Basic Compression, ausschließlich für Bulk Loads implementiert.

Innerhalb von HCC gibt es unterschiedliche Komprimierungsverfahren. Diese können durch die Schlüsselwörter ARCHIVE und QUERY mit dem Zusatz LOW bzw. HIGH angezeigt werden. Bei typischer Warehouse Verwendung mit Low Concurrency empfiehlt sich der Einsatz von QUERY. Wie das Schlüsselwort ARCHIVE schon anzeigt, sollten Daten, die nicht mehr verändert werden und für eine Langzeitspeicherung vorgesehen sind, mit ARCHIVE HIGH gespeichert werden. Die Komprimierungsverfahren unterscheiden sich intern in der Verwendung unterschiedlicher Compression Algorithmen und unterschiedlich großer Logical Compression Units. Wie zu erwarten, liefert die Compression Ratio bei der Verwendung von ARCHIVE höhere Werte als bei QUERY. Folgende Beispiele zeigen die Verwendung der Syntax. Auch hier wie bei der BASIC und OLTP Compression hat sich die Syntax in 12*c* verändert.

HCC mit Query (Default ist HIGH) 12*c:*

```
CREATE TABLE mass(...) COLUMN STORE COMPRESS FOR QUERY[LOW|HIGH]
```

HCC mit Archive (Default ist LOW) 12*c:*

```
CREATE TABLE mass(...) COLUMN STORE COMPRESS FOR ARCHIVE[LOW|HIGH]
```

Abgesehen von den strukturierten Daten stellt sich natürlich auch die Frage nach der Komprimierung von **unstrukturierten Daten.** Unstrukturierte Daten vom Datentyp XML, CLOB und BLOB sind in der Regel sehr speicherintensiv. Aus diesem Grunde ist eine Komprimierung dieser Daten sinnvoll. Eine Möglichkeit wäre die Nutzung des Package **UTL_COMPRESS** , das mit Oracle Version 10*g* eingeführt worden ist. Mit UTL_COMPRESS lassen sich RAW, BLOB oder BFILE Daten komprimieren bzw. de-komprimieren. Das Resultat von UTL_COMPRESS entspricht dabei dem der Werkzeuge *compress* in Unix Umgebungen und *zip* in Windows Umgebungen. UTL_COMPRESS erfordert allerdings eine vollständige Programmierung in PL/SQL für die Komprimierung und die De-Komprimierung der Daten und ist unabhängig von der Speicherung in der Datenbank.

Das Ziel LOB **Komprimierung in der Datenbank ohne zusätzliche Programmierung** bereitzustellen, ist mit Oracle Database 11*g* und der Einführung des neuen Datentyps SECUREFILE für Large Objects implementiert worden. Eine Eigenschaft von Oracle SecureFiles ist dabei die Möglichkeit, Komprimierung einzuschalten. Dies erfordert allerdings den Einsatz der Advanced Compression Option. Dabei sind folgende Einstellungen bei der Komprimierung möglich:

- DEDUPLICATE: LOBs mit identischem Inhalt werden physikalisch nur einmalig gespeichert. Diese Einstellung ist besonders sinnvoll bei der Nutzung von großen LOBs, die mehrfach gespeichert werden wie z.B. Email Attachments.
- COMPRESS HIGH (bzw. MEDIUM, LOW): Reduzierung des Speicherbedarfs von LOBs durch Komprimierung. Diese Komprimierung wird durch einen Standardalgorithmus durchgeführt und kann wahlweise mit einer hohen, mittleren bzw. niedrigen Komprimierungsrate durchgeführt werden. Die LOB-Komprimierung mit Parametereinstellung HIGH hat dabei einen höheren CPU-Bedarf als die Komprimierung der LOBs mit der SecureFile Standardkomprimierung MEDIUM oder mit der Komprimierung LOW.

Wichtig zu wissen ist, dass die LOB-Komprimierung unabhängig von der Tabellenkomprimierung ist und beim CREATE TABLE oder ALTER TABLE separat über die SecureFile LOB-Storage-Klausel angegeben werden muss.

```
CREATE TABLE nachrichten_text (dok_id NUMBER,...,text_info CLOB)
LOB (text_info)
STORE AS SECUREFILE(DEDUPLICATE COMPRESS HIGH DISABLE STORAGE IN ROW)
```

Je nach gespeichertem Format HTML, Text, ASCII, PDF, GIF usw. sind die **Komprimierungsraten** unterschiedlich. Zweistellige Werte bei der Komprimierungsrate können dabei allerdings nichts Ungewöhnliches sein. Speziell bei I/O intensiven Applikationen kann der Einsatz von Komprimierung von Vorteil sein. Umgekehrt macht es keinen oder wenig Sinn, schon vorkomprimierte Formate zu komprimieren.

Neu in 12*c* ist übrigens auch die **Komprimierung bei der Übermittlung von Daten über SQL*Net** zwischen Client und Server. Die Konfiguration erfolgt dabei über die Einstellung der neuen SQL*Net Parameter SQLNET.COMPRESSION zur Aktivierung, SQLNET.COMPRESSION_LEVELS zur Einstellung der Level und SQLNET.COMPRESSION_THRESHOLD zur Angabe einer unteren Grenze. Die Verwendung kann dabei auf verschiedenen Ebenen wie Connection (z.B:. connect string), Service (tnsnames.ora, ldap.ora) oder Datenbank (sqlnet.ora) statt finden. Auch hier ist die Voraussetzung die Lizenzierung der Advanced Compression Option.

Zum Abschluss vielleicht noch eine paar Informationen zur **Indexkomprimierung:** Indizes werden standardmäßig komprimiert, sobald sie als Bitmap Index angelegt werden. Wegen des speziellen Lockings bietet sich die Verwendung allerdings nur bei lesenden Zugriffe an. Darüber hinaus gibt es allerdings seit jeher auch eine Komprimierung für B*tree Indizes und IOTs - die sogenannte **Index Key Compression**. Das Prinzip der Index Key Compression beruht dabei auf der Eliminierung von sich wiederholenden Schlüssel-Werten (auch Präfix genannt) eines **nonunique single column**- oder eines **unique multicolumn**- Index. Ob die Indizes sich für die Art der Komprimierung eignen, lässt sich beispielsweise relativ einfach über das Kommando ALTER INDEX VALIDATE STRUCTURE prüfen. Das Vorgehen und die Funktionsweise ist nachzulesen in vielen Blogs und Artikel und soll daher hier nicht ausgeführt werden.

RMAN, Compression Advisor und Data Pump

Wie kann man herausfinden, welche Komprimierungsrate zu erwarten ist? Kann die Komprimierung beim Verlagern von Daten aktiviert werden? Wie können Backup Daten komprimiert werden. Dies sind einige der häufig gestellten Fragen im Zusammenhang mit Compression.

Um den Grad der Speicherplatzeinsparung festzustellen, ist es naheliegend, neue Segmente mithilfe der neuen Speichereinstellung zu erstellen und dann den Quotient aus nicht komprimierten und komprimierten Objekten - die sogenannte Compression Ratio - zu berechnen. Eine Alternative ist die Nutzung des **Compression Advisors**. Seit Oracle Database 11*g* Release 2 steht standardmäßig der Compression Advisor in der Datenbank zur Verfügung. Ohne zusätzliche Installation ist dieser Advisor über das Package DBMS_COMPRESSION **in jeder Edition** sofort einsetzbar. Realisierte Compression Projekte zeigen übrigens, dass die Berechnung der Ratio durch den Compression Advisor sehr realistische Werte und Annäherungen zur tatsächlichen Speicherung liefert. Hat man übrigens noch keinen Zugriff auf eine 11*g* Release 2 Installation, kann man Unterstützung durch eine zusätzliche Package-Installation erhalten. Download und Nutzungsbeschreibung dazu finden sich auf OTN (siehe Informationen).

Folgende Fragestellungen können mit DBMS_COMPRESSION gelöst werden.

- Sind alle Blöcke meiner Tabellen komprimiert? Und welcher Komprimierungstyp wurde verwendet? Die Antwort liefert die Prozedur GET_COMPRESSION_TYPE. Über die einfache Eingabe einer ROWID kann der Advisor ermitteln, ob und welche Komprimierung verwendet wurde.

- Welche Komprimierung (Ratio) kann beim Einsatz der unterschiedlichen Komprimierungstypen erwartet werden? Hier kann die Prozedur GET_COMPRESSION_RATIO weiterhelfen. Nach Eingabe einer Komprimierungsart wird die Ratio des Segments ermittelt. Zusätzlich zu den Komprimierungstypen OLTP und BASIC können auch HCC Komprimierungstypen wie QUERY LOW, QUERY HIGH, ARCHIVE LOW und ARCHIVE HIGH **ohne** Zugriff auf ein Exadata Storagesystem getestet werden.

Auch Indizes und unstrukturierte Daten nehmen einen Anteil am gesamten Speicherplatz ein und sollten in die Berechnung einfließen. Der Compression Advisor liefert allerdings dazu keine Unterstützung in 11*g* Release 2. Ab 12*c* gibt es allerdings zusätzlich eine Erweiterung für unstrukturierte Daten. Der erweiterte Aufruf sieht dann folgendermaßen aus:

```
  DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
SCRATCHTBSNAME       => 'USERS',
TABOWNER             => 'SH',
TABNAME              => 'BASIC_LOB',
LOBNAME              => 'TEXT',
PARTNAME             => '',
COMPTYPE             => 128,
BLKCNT_CMP           => b_cmp,
BLKCNT_UNCMP         => b_uncmp,
LOBCNT               => lob_cnt,
CMP_RATIO            => cmp_ratio,
COMPTYPE_STR         => cmp_str);
```

COMPTYPE steht hierbei für den überprüften Komprimierungstyp – 128 bedeutet dabei HIGH. Die Liste aller Komprimierungstypen findet sich im Handbuch. Nach Ablauf des vollständigen Skripts kann das Ergebnis dann folgendermaßen aussehen.

```
Sampling percent: 2.5
Uncomp blocks: 1246  Comp blocks: 638
Number of lobs sampled: 4980
compression ratio: 1.9
```

Übrigens auch der **Segment Advisor** kann in gewissen Fällen eine Unterstützung liefern. Beim Automatic Segment Advisor Lauf werden auch OLTP Compression Empfehlungen für Tabellen gegeben. Diese müssen allerdings bestimmten Eigenschaften genügen – nämlich mindestens 10 MB groß sein und mindestens 3 Indizes aufweisen. Es schadet also nicht, einen Blick auf das Ergebnis der automatischen Maintenance Task (meist im nächtlichen Lauf) zu werfen.

Backups können seit Oracle Database 10*g* mit einer speziellen **Backup Compression** - BZIP2 Algorithmus - durchgeführt werden. Backupsets werden dann, bevor sie auf die Platte geschrieben werden, komprimiert. Bei der späteren Nutzung dieses Backups ist allerdings kein zusätzlicher separater De-Komprimierungsschritt mehr notwendig. Folgende Syntax zeigt die Anwendung von Komprimierung beim RMAN Backup.

```
RMAN> BACKUP AS COMPRESSED BACKUPSET DATABASE;
```

Allerdings kann die Anwendung von Komprimierung die Dauer des Backups um ein Vielfaches verlängern. Aus diesem Grund wurden ab 11*g* mit der Advanced Compression Option weitere Algorithmen zur Verfügung gestellt, um die Geschwindigkeit beim Backup zu erhöhen. In 11*g* Release 1 handelte es sich dabei um den ZLIB Algorithmus. Ab 11g Release 2 änderte Oracle seine Strategie und veröffentlicht nicht mehr die Namen der verwendeten Algorithmen. Die Bezeichnungen der Algorithmen lauten nun BASIC, LOW, MEDIUM und HIGH. Einstellbar ist der gewählte Algorithmus über das folgende RMAN Kommando. Die Default Einstellung ist BASIC, die der 10*g* Variante entspricht.

```
RMAN> CONFIGURE COMPRESSION ALGORITHM 'BASIC|LOW|MEDIUM|HIGH';
```

Wie viel schneller ein Backupset komprimiert bzw. recovered werden kann, hängt von der Umgebung und dem gewählten Algorithmus ab. Falls ein I/O-Bottleneck vorliegt aber CPU zur Verfügung steht, kann der Algorithmus HIGH zu guten Ergebnissen führen. HIGH verwendet mehr CPU bei hoher Platzeinsparung und vermindert somit die Anzahl der I/Os zum Schreiben des Backups. Auf der anderen Seite kann bei einer Begrenzung der CPU Ressource der Einsatz von LOW oder MEDIUM sinnvoller sein. Es zeigt sich, dass mit diesen Algorithmen Platzeinsparungen bis zu 80% sogar bei höherer Backup Performance (ca. 60%) erzielt werden können. Weitere Ergebnisse aus Kunden POCs finden sich im White Paper „Oracle Advanced Compression Helps Global Fortune 500 Company“ (siehe Weitere Informationen). Komprimierte Daten in der Datenbank werden übrigens bei Einsatz der RMAN Komprimierung weiter minimiert. Der Einsatz kann also auch in diesem Fall sinnvoll sein.

Auch der **Data Pump Export** profitiert von Komprimierungsmöglichkeiten. Bislang wurden automatisch nur die Metadaten komprimiert; mit 11*g* und der Advanced Compression Option können nun auch die Tabellendaten komprimiert werden. Beim Data Pump Export unter Angabe der Option *compression=all* werden automatisch die Tabellendaten komprimiert; beim Import ist keine weitere Angabe bzw. kein weiterer De-Komprimierungsschritt nötig. Dabei bleibt die Komprimierung vollständig Applikations-transparent, und somit gibt es keine Einschränkungen bei der Data Pump Funktionalität. Auch hier sind Komprimierungen von 75% keine Besonderheit und können mit Werkzeugen wie GNU *zip* verglichen werden. Neu mit 12*c* ist die Möglichkeit, diese Einstellung zu ändern. Unabhängig von der Einstellung im Export bzw. von der Einstellung im Tablespace der Zieldatenbank können somit Tabellen mit eigenen Compression Einstellungen erzeugt werden. Dazu ist ein neuer Metadaten TRANSFORM Parameter nötig. Folgendes Beispiel zeigt eine Implementierung. Der Tablespace der Zieldatenbank besitzt dabei keine Compression Einstellung. Die Tabelle wird im folgenden Beispiel mit Basic Compression importiert.

```
impdp dumpfile=sh.dmp directory=home tables=sh.cust_copy
TRANSFORM = TABLE_COMPRESSION_CLAUSE:\"COMPRESS BASIC\"
```

Die Compression Klauseln (siehe TABLE_COMPRESSION_CLAUSE) entsprechen dabei der Tabellen Compression Klauseln im CREATE oder ALTER TABLE Kommando.

Fazit und Ausblick

Komprimierung ist in der Datenbank ein nicht mehr wegzudenkendes Feature, das in jedem Release weiterentwickelt wird. Generell muss man wissen, dass die Komprimierung dabei ohne zusätzliche Installation oder Aktivierung in der Datenbank zur Verfügung steht - muss nicht nachträglich installiert oder eingeschaltet werden. Möchte man bestimmte Feature verwenden, ist die Implementierung durch die vorgegebene Syntax erforderlich. Im Umkehrschluss bedeutet dies aber auch, dass Compression beispielsweise über die Algorithmen der Advanced Compression Option nicht ausgeschaltet werden kann. Es gilt der Grundsatz: Werden die entsprechenden Funktionen nicht

verwendet, ist auch keine Lizenzierung erforderlich – wie übrigens auch im Fall von anderen Optionen (z.B. bei Partitioning Option).

Um herauszufinden, ob Komprimierung in der eigenen Umgebung sinnvoll ist, sollte unbedingt im ersten Schritt der Compression Advisor bei strukturierten Daten genutzt werden. Ab 12*c* bietet dieser sogar ein Interface für Large Objects. Zusätzliche Tests sollten allerdings immer eingeplant werden, da es zu Änderungen an Ausführungsplänen, Query und DML Performance kommen kann.

Vergleicht man die Feature über die Jahre kann man feststellen, dass der Fokus bis 11*g* auf der Einführung von verschiedenen Komprimierungsalgorithmen für unterschiedliche Daten lag; in der aktuellen Version 12*c* Release 1 hingegen hat eine Erweiterung auf der Automatisierung und der Kategorisierung der Daten stattgefunden. Bleibt abzuwarten, was im kommenden Release implementiert wird.

Weitere Informationen

- **My Oracle Support Notes:**
  - Script to investigate a b-tree index structure (DOC ID **989186.1)**
  - Master Note for OLTP Compression (Doc ID 1223705.1)
  - Using the new UTL_COMPRESS Oracle Supplied Package (Doc ID 249974.1)

- OTN Download für Oracle Advanced Compression Advisor: http://www.oracle.com/technetwork/database/options/compression/compression-advisor-095705.html

- Oracle Dojo: Komprimierung in der Datenbank

- Deutschsprachige Tipps der DBA Community: http://blogs.oracle.com/dbacommunity_deutsch

- **Oracle White Paper:**
  - Oracle Advanced Compression Helps Global Fortune 500 Company
  - Oracle Advanced Compression with Oracle Database 12*c*

- **Handbücher:**
  - VLDB and Partitioning Guide
  - SQL Language Reference
  - PL/SQL Packages and Types Reference
  - Database Licensing Information

Kontaktadresse

Ulrike Schwinn
Oracle BU DB – Business Unit Database

ORACLE Deutschland B.V. & Co. KG
Riesstr 25, 80992 München

Telefon: +49 89 1430 1865
E-Mail Ulrike.Schwinn@oracle.com
Internet: http://blogs.oracle.com/dbacommunity_deutsch