Space - the final frontier:
Speicher- und Performanceaspekte in Oracle Tablespaces

Martin Hoermann
ORDIX AG
Paderborn

Schlüsselworte

Block, Extent, Segment, Tablespace, Space, Füllgrad

Einleitung

In diesem Vortrag werden die zahlreichen Aspekte des physikalischen Speichermanagements von Oracle beleuchtet. Der Beitrag beantwortet die Fragen, wie der Füllgrad von Tablespaces zu berechnen ist, welche Unterschiede es bei den verschiedenen Arten gibt und warum die Trennung von Tabellen und Indizes nicht mehr zeitgemäß ist. Weiterhin wird die Frage diskutiert, wie viele Tablespaces in einer Datenbank notwendig sind.

Ebenso untersuchen wir die internen Strukturen von Tabellen und zeigen, wie sich der Füllgrad einer Tabelle bestimmen und beeinflussen lässt. Dieses Wissen ist wesentlich, um den Sinn und Unsinn von terrabyte-großen Tabellen zu hinterfragen.

Tablespaces, Segmente, Extents und Blöcke

Ein Tablespace ist ein Container für Segmente einer Oracle-Datenbank. Ein Segment wiederum ist ein Objekt, welches physikalischen Speicherplatz beansprucht, also z.B. eine Tabelle, einen Index, eine Partition oder einen Cluster. Wie der Name bereits verrät gehören auch UNDO- und temporäre Segmente hierzu.

Ein Segment besteht aus einem oder mehreren Extents. Ein Extent besteht aus einer Menge von physikalisch hintereinanderliegenden Blöcken. Auf der einen Seite dienen Extents dazu, die Verkettung zusammengehöriger Blöcke eines Segments zu vereinfachen. Auf der anderen Seite dienen Extents dazu die Speicherplatzanforderungen von Segmenten auf unterschiedliche physikalische Dateien zu ermöglichen. So war es schon zu Zeiten von Dateisystemen mit maximal 2 GB Dateisystemgröße selbstverständlich möglich, Tabellen anzulegen die deutlich größer waren. Da die Blöcke eines Extent physikalisch hintereinanderliegen, muss ein Extent zwangsläufig in genau einer Datendatei liegen.

Ob die Blöcke eines Extent physikalisch auf einem Datenträger hintereinanderliegen ist aufgrund von zahlreichen Abstraktionsschichten wie LUNs und Striping nicht gewährleistet. Aus Sicht der Datei und des Dateisystems lässt sich das aber vereinfachend annehmen.

Tablespaces und Datendateien

Die Größe eines Tablespace bestimmt sich durch die Größe der ihm zugeordneten Datendateien. Jede Datendatei wird mit einer initialen Größe angelegt. Optional hat die Datendatei die Eigenschaft autoextensible, mit der sich eine automatische Vergrößerung in definierten Stücken bis hin zu einer maximalen Größe konfigurieren lässt.

Die Zuordnung von Extents zu Datenbankdateien übernimmt Oracle automatisch. Eine manuelle Zuordnung ist zwar möglich, jedoch seit dem Ende des Oracle Parallel Servers eher ungebräuchlich. Ob eine Datei in einem herkömmlichen Dateisystem oder in ASM liegt ist aus Sicht des Speicherplatzverbrauches weitestgehend irrelevant.

Datendateien, High Water Marks und Fragmentierung

Datendateien beginnen mit einem Header der meist 64 KB oder 1 MB groß ist. Dahinter wird der Platz nach und nach mit Extents gefüllt. In jeder Datei zeigt eine High Water Mark an, bis zu welcher höchsten Position in der Datei jemals ein Extent lag. Wird ein Extent oberhalb der High Water Mark angelegt, so wird diese um die Größe des Extents nach oben verschoben. Datendateien können nur bis zum am weitesten hinten liegenden Extent verkleinert werden.

Durch das Löschen oder die Reorganisation von Segmenten entstehen Lücken in der Datendatei, die durch neue Extents gefüllt werden können. Das Resultat wird auch Fragmentierung genannt. Passen Lücken und neue Speicherplatzanforderungen gut zusammen ist die Fragmentierung i.d.R. unkritisch. Dies wird heute durch standardmäßige Locally Managed Tablespaces erreicht, bei denen alle Extents 8, 128, 1024 usw. Blöcke groß sind. Alle Lücken sowie der Bereich oberhalb der High Water Mark verwaltet die Datenbank als freie Extents.

Der Füllgrad der Tablespaces nimmt natürlich durch viele „Löcher“ ab, aus Sicht des Dateisystems wird der Platz allerdings nicht wieder freigegeben. Beim Backup mit dem Recovery Manager werden diese Löcher mit gesichert, dafür muss der Recovery Manager nicht die gesamte Datei sondern nur den Bereich bis zur High Water Mark berücksichtigen.

UNDO und temporäre Tablespaces

Im Gegensatz zu permanenten Tablespaces beinhalten Tablespaces vom Typ UNDO und TEMP nur Extents, die zur Laufzeit benötigt werden. Die Extents werden automatisch von der Datenbank allokiert und wieder freigegeben. Aber auch hier fällt die High Water Mark der zugehörigen Dateien nicht zurück. Bei Backups werden temporäre Tablespaces übersprungen. Mit ein paar administrativen Mitteln ist es möglich, die Tablespaces im laufenden Betrieb zu wechseln und somit den Platzverbrach zu reduzieren, wenn diese zur Laufzeit einmal besonders groß geworden sind.

Trennung von Daten und Indizes

Aus Sicht des I/O-Systems gibt es bei Tablespaces nur zwei Formen von lesenden Zugriffen. Dies sind Singleblock-I/Os und Multiblock-I/Os. Während die ersten meist aus dem Lesen über ROWID resultieren, entstammen Multiblock-I/Os aus Full-Table-Scans. Bei Multiblock-I/Os sollte die Mehrzahl der betroffenen Extents größer als die maximalen I/O-Size sein, da ein I/O maximal bis zum Ende des Extents geht. Bei einer Extent-Größe von 64 KB und einer maximalen I/O-Größe von 1 MB sind gegenüber einem 1 MB Extents 15 I/O-Operationen zusätzlich notwendig. Weiterhin ist natürlich wichtig, dass die I/O-Aufträge über möglichst viele Platten verteilt werden. Daraus resultiert der Ansatz „stripe and mirror everything“.

Da heute zwischen dem Dateisystem und den physikalischen Festplatten nahezu immer eine oder mehrere Abstraktionsschichten wie LUNs und Striping liegen, sind die I/O-Charakteristika einzelner Dateien in der Regel gleichartig. Aus diesem Sachverhalt erschließt sich keine vernünftige Begründung warum heute Daten von Indizes in unterschiedliche Tablespaces getrennt werden sollten.

Füllgrad von Tablespaces

Der aktuell maximale Platz eines Tablespace berechnet sich aus der Summe der Bytes der zugehörigen Daten- oder temporären Dateien. Der maximal mögliche Platz berechnet sich, unter Berücksichtigung der Eigenschaft der automatischen Erweiterbarkeit, aus dem freien Platz im Dateisystem und der maximalen Größe der Datendateien. Der belegte Platz berechnet sich aus dem Datenbankdatei-Header und den belegten Extents.

Wie viele Tablespaces

Die folgenden Tablespaces sind für jede produktive Datenbank obligatorisch:

- SYSTEM
- SYSAUX
- TEMP
- UNDO

Darüber hinaus benötigen die Anwendungen auf einer Datenbank lediglich einen Tablespace! Von dieser Regel sollte nur Abstand genommen werden, wenn dafür nachvollziehbare Gründe vorliegen. Denkbar sind dabei:

- sehr viele Read-Only-Objekte, die in ein Read Only Tablespace ausgelagert werden, um die Backup-Zeiten zu reduzieren
- Datenbewirtschaftung mit Transportable Tablespaces
- bei sehr große Datenmengen, um den Restore einzelner Datenbankdateien zu beschleunigen

In der Praxis lässt sich fast immer beobachten, dass die Summe des vorgehaltenen Datenvolumen für das Wachstum von Tablespaces nahezu proportional mit der Anzahl der Tablespaces wächst. Ihr Hersteller von Speichersystemen wird sich darüber freuen.

Wie ein Segment entsteht

Ein Segment entsteht in der Regel durch das Erstellen eines Objektes, welches eigenen Speicherplatz benötigt. Die Erstellung einer View führt beispielsweise nicht zur Erstellung eines Segments, da lediglich die Definition im Data Dictionary abgelegt wird. Eine Materialized View „materialisiert“ die Ergebnismenge, benötigt daher eigenen Speicherplatz und führt zu der Erstellung eines Segments. Genauso ist es mit Tabellen, Indizes, Cluster, UNDO- und temporären Segmenten. Ist ein Objekt partitioniert, so bildet jede Partition bzw. Subpartition ein einzelnes Segment.

Mit Oracle 11g ist es möglich, Objekte mit der Option „Deferred Segment Creation“ zu erstellen. So wird das Segment, welches immer aus einem initialen Extent (i.d.R. 8 Blöcke) besteht, erst erzeugt, wenn der erste Datensatz eingefügt wird. Dies hat übrigens den interessanten Begleiteffekt, dass bei Sequenzen die erste Sequenznummer „verloren“ geht.

Extent Map

Zur Reduzierung der Verwaltungsinformationen und aus Performance-Gründen werden eine Menge hintereinanderliegender Blöcke in so genannten Extents zusammengefasst. In Locally Managed

Tablespaces (LMT) sind die ersten 16 Extents 8 Blöcke groß, die nächsten Gruppen von Extents sind 128, 1024 usw. Blöcke groß. In einem LMT Tablespace Uniform Extent Size kann der DBA eine gleichförmige Extent-Größe für alle Extents angeben.

Wird nun neuer Platz benötigt, wird ein neues Extent im Tablespace angelegt und dem Segment zugeordnet. Die Extents bilden die äußere Struktur des Segments. In den nächsten Abschnitten beleuchten wir nun die innere Struktur.

Aufbau des Segments

Der Aufbau eines Segments besteht im Wesentlichen aus den folgenden Typen. Dabei ist jeder Oracle-Block von genau einem Typ:

- Segment Header: Dieser enthält beispielsweise Informationen zur High Water Mark und zu den Freispeicherlisten.
- Freispeicherlisten: In Tablespaces mit Automatic Segment Space Management (ASSM) sind dies sogenannte Bitmap Freelists, wird der Tablespace manuell verwaltet sind dies Freelists und Freelist Groups.
- Data Blocks: In diesen Blöcken befinden sich die eigentlichen Daten. Daten können in diesem Fall auch Index oder UNDO-Daten sein.
- Unformatted Blocks: Am Ende des Segments kann es noch nicht-formatierte Blöcke geben, diese werden erst bei Bedarf zu Daten- oder Freispeicherlisten-Blöcken formatiert.

Werden neue Daten mittels INSERT in eine Tabelle eingefügt, so wird über die Freispeicherlisten ein passender Block ausgewählt. Mit dem heute üblichen ASSM wird die freie Kapazität in 25 %-Schritten verwaltet. Steht kein freier Platz mehr zur Verfügung, so wird hinter der HWM neuer Platz geschaffen, in dem unformatierte Blöcke formatiert werden. Stehen keine unformatierten Blöcke zur Verfügung so wird ein neues Extent allokiert.

Die Freispeicherlisten können mit dem so genannten APPEND-Hint umgangen werden. Bei dem daraus folgenden Direct Path Inserts werden die Datensätze direkt hinter der High Water Mark eingefügt. Dies ist für große Datenmengen einerseits performanter, führt jedoch anderseits möglicherweise zu nicht genutzten Platz innerhalb des Segments.

Aufbau eines Datenblocks vom Typ Data

Ein Datenblock besteht aus den folgenden Bereichen:

- Block Header: Hier finden sich allgemeine Verwaltungsinformationen wie beispielsweise das zugehörige Segment, die Speicherplatzadresse und die aktuelle SCN-Nummer.
- Row Directory: Hier wird jede Zeile und jedes Feld des Datenbereichs adressiert. An dieser Stelle entscheidet sich durch die Längenangabe des benötigten Speicherplatzes, ob ein Feld den Wert NULL enthält. Darüber hinaus benötigt der Wert NULL keinen weiteren Speicherplatz. Mehrere NULL-Werte am Ende einer Zeile benötigen lediglich ein Byte im Row Directory.
- ITL-Slots: Ändert eine Transaktion Daten im Block so belegt sie einen ITL-Slot. Im Row Directory wird für jede geänderte Zeile eine Referenz auf den Slot eingetragen. Weiterhin enthält der ITL-Slot Verweise zu den UNDO-Informationen der Transaktion.

- Daten: Hinter den Verwaltungsinformationen kommen die Daten als Byte-Strom. Feldtrenner werden nicht benötigt, da die Adressierung durch das Row Directory jedes Feld eindeutig bestimmen kann.

- Block Footer: Hier befindet sich erneut die SCN Nummer. Unterscheiden sich SCN-Nummer im Header und im Footer, so ist der Block inkonsistent. Dies kann beispielsweise bei Online-Backups passieren. Der Recovery Manager liest den Block in diesem Fall einfach erneut.

Der belegte Platz im Datenbereich bestimmt sich erstens durch die gewählten Datentypen und zweitens durch die Dateninhalte. Der Wert NULL benötigt bis auf den Verweis im Row Directory keinen Speicherplatz. Dies gilt für alle Datentypen, insbesondere auch für CHAR, der ansonsten immer eine fixe Länge hat.

Bei einigen Datentypen ist die Anzahl benötigter Bytes immer gleich, z.B. DATE, TIMESTAMP, CHAR, bei anderen Datentypen ist die Anzahl der Bytes abhängig von deren Inhalt, z.B. NUMBER, VARCHAR2. Eine sorgfältige Wahl der Datentypen und der Dateninhalte beeinflusst also ganz wesentlich den Platzverbrauch einer Tabelle.

Werden Datensätze in einem Block aktualisiert (UPDATE) so kann der Speicherplatz steigen oder sinken. Damit es bei einer Vergrößerung nicht sofort zu Speicherengpässen und daraus resultierenden Chained Rows kommt gibt es in jedem Block einen reservierten Speicherplatz (PCT_USED). Der Default-Wert von 10 % ist insbesondere für große Tabelle sorgfältig zu prüfen.

Für detaillierte Analysen kann ein Block mit Hilfe des folgenden Kommandos in das Diagnostic Verzeichnis (user_dump_dest) geschrieben werden. Die Dateinummer und die Block-ID lassen sich mit Hilfe der View dba_extents ermitteln.

```
ALTER SYSTEM DUMP DATAFILE 4 BLOCK 259;
```

In der Regel sind die ersten beiden Blöcke des ersten Extent Freispeicherlisten und der dritte Block beinhaltet den Segment Header.

Platzverbrauch messen

Der Platzverbrauch eine Blocks lässt sich mit dem Paket dbms_space ermitteln. Um eine korrekte Ausgabe sowohl für unterschiedliche Tablespace Verwaltungen (Locally vs. Manual Managed) als auch für unterschiedliche Speicherverwaltungen (Automatic vs. Manual Segment Space Management) zu erhalten, sei an dieser Stelle auf den Blog von Tom Kyte verwiesen.

Um den Inhalt einzelner Felder genauer zu untersuchen bietet Oracle die Funktion `dump` an. Mit ihr wird sowohl der Datentyp (Typ) als auch die Länge in Bytes (Len) ausgegeben. So hat beispielsweise ein gefülltes Feld vom Typ Timestamp(6) vier Byte mehr als ein Date-Feld mit sieben Bytes. Das entspricht immerhin 57 Prozent mehr Datenvolumen.

```
dump( <date> )   : Typ=12  Len=7: 120,11...

dump( <timestamp): Typ=180 Len=11: 120,1....
```

Wer sich tiefergehend mit den hier diskutierten Themen beschäftigen will sei auf den Concept Guide in der Oracle-Dokumentation oder auf Tom Kytes Expert Oracle Database Architecture verwiesen.

Von roten Riesen und weißen Zwergen - ein Fazit

Tabellengrößen von einem Terabyte und mehr sind heute Realität. Inwiefern das immer notwendig ist, hängt maßgeblich vom logischen und physikalischen Tabellendesign ab. Steht das Layout einer Tabelle fest, so lässt sich der Speicherverbrauch auf den einzelnen Ebenen analysieren. Die einzelnen Schritte wurden aufgeführt:

- Größe einzelner Datenfelder mit der SQL-Funktion `dump`
- der Füllgrad eine Datenblocks mit `dbms_space`
- das Layout des Segments über `dba_segments` und `dba_extents`

Im Vortrag wird aufbauend auf den hier vorgestellten Themen auch die Kompression von Segmenten behandelt.

Kontaktadresse:
Martin Hoermann
ORDIX AG
Westernmauer 12-16
D-33098 Paderborn

Telefon: +49 (0) 5252 - 10630
Fax: +49 (0) 12-345 6788
E-Mail info@ordix.de
Internet: www.ordix.de