Tablespaces und Datendateien

Thomas Klughardt
Quest Software
Köln

Schlüsselworte:

Tablespace, Datendatei, Planung, Eigenschaften, Best Practices

Einleitung

Eine wichtige Aufgabe beim Anlegen einer Datenbank ist die Aufteilung der Tablespaces, die wiederum auf physikalische Datendateien abgebildet werden. Darüber sollte man sich am besten Gedanken machen, bevor die Anwendung sich in der Datenbank breit macht.
Hier geht es um grundsätzliche Gedanken zum Layout der Datenbank, um das Klären der Begrifflichkeiten, und um Überlegungen und Best Practices zum Gestalten der Tablespaces.

Definition - Was sind Tablespaces und Datendateien?

Ein Tablespace ist ein logischer Container in einer Datenbank, der Objekte, genauer Segmente, enthält. Diesem logischen Container sind jeweils ein oder mehrere physische Container, die Datendateien, zugeordnet. Deshalb kann man sagen, dass Tablespace und Datendatei die Brücke zwischen logischer und physischer Struktur einer Datenbank bilden.

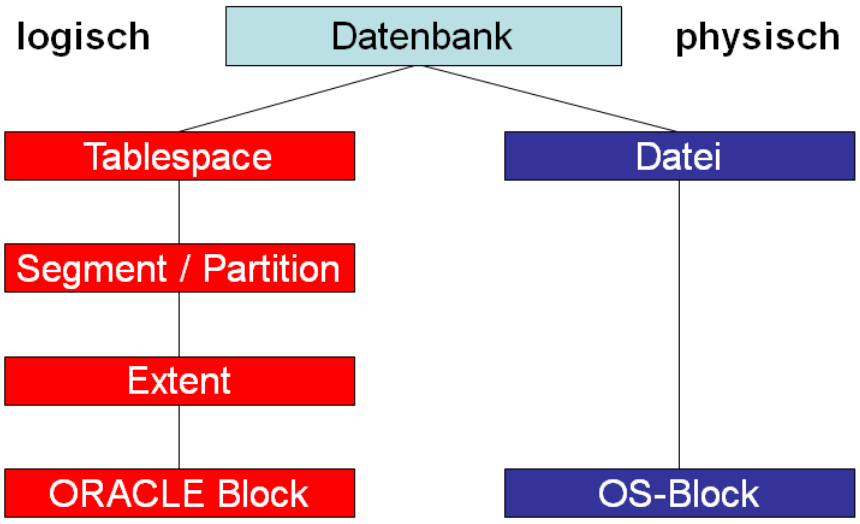

Abb. 1: Tablespaces sind der logische, Datendateien der physische Teil einer Datenbank

Der Grund für diese Brücke ist, dass die Oracle Datenbank zwar auf verschiedenen Betriebssystemen verfügbar ist, man aber nichts davon merken soll. Die Basisadministration ist natürlich betriebssystemabhängig, für die Anwendung und das Schemamanagement soll das Betriebssystem

jedoch unerheblich sein. Ein Script zum Anlegen von Schemaobjekten soll unter Unix genauso funktionieren wie unter Windows. Deshalb wurde diese Abstraktionsschicht eingeführt.

Planung - Warum sollte man frühzeitig darüber nachdenken?

Es ist sinnvoll sich frühzeitig Gedanken über die Aufteilung der Tablespaces zu machen und sich dafür ruhig auch Zeit zu nehmen und mit den Anwendungsarchitekten zu sprechen. Während man um Instanzparameter zu ändern, oder bestimmte Datenbankfeatures ein- und auszuschalten schlimmstenfalls einmal die Instanz durchstarten muss, ist ein Ändern der Tablespaces mit erheblich mehr Aufwand und unter Umständen auch mit mehr Auszeit verbunden. Die einzige meist noch aufwändigere Aktion ist das Ändern des Zeichensatzes.

Deshalb folgen hier einige Punkte, über die man sich bei der Aufteilung Gedanken machen sollte.
Eine gute Möglichkeit Tablespaces zu nutzen ist die logische Aufteilung der Anwendung oder die Aufteilung nach Zuständigkeiten. Für die Anwendung spielt es keine Rolle, in welchem Tablespace eine Tabelle liegt, für die Administratoren schon. Besonders pflegeintensive Objekte, wie zum Beispiel schnell wachsende Tabellen, können deshalb in einen Tablespace gelegt werden, dem man mehr Aufmerksamkeit schenkt.
Auch für die Granularität von Backups und Restores sind Tablespaces wichtig. Mit RMAN kann man zwar einzelne Blöcke wiederherstellen, einfacher ist es aber eine Restore und Recovery eines bestimmten Tablespace durchzuführen. Sind die dann sinnvoll gewählt, kann der Rest der Datenbank solange weiterarbeiten und ein Restore/Recovery eines Tablespaces geht natürlich schneller, als das der gesamten Datenbank.
Eine beliebte Aufteilung ist auch noch die Trennung kleiner und großer Objekte, und von Tabellen und Indizes. Das hat meist historische Gründe, findet ein Striping und Mirroring über alle Platten statt, ist es für die Performance eher nebensächlich. Interessanter wäre es dann, Objekte zu gruppieren, die schnell fragmentieren und öfter reorganisiert werden müssen. Wenn aber nicht nach dem „Stripe and Mirror Everything“-Prinzip (SAME) vorgegangen wird, ist es natürlich schon interessant besonders IO intensive Objekte auf schnellere Platten zu legen.
Ein oft vernachlässigter aber sehr wichtiger Punkt ist die Sicherheit. Usern lassen sich gezielt Quotas auf Tablespaces geben. Auch wenn der Anwender nichts an den Basisobjekten einer Anwendung ändern darf, so kann er doch die Anwendung selbst zum Stillstand bringen, indem er einfach deren Tablespace vollschreibt. Das kann man durch verschiedene Tablespaces und Tablespace Quotas verhindern.

Zusätzlich gibt es inzwischen auch sehr nützliche Tablespace-Features wie Transportable Tablespace, mit dem man einen Metadaten und Datendateien eines Tablespaces in eine andere Datenbank kopieren kann. Der Tablespace muss für die Zeit zwar im Read Only Modus sein, es geht aber wesentlich schneller als mit Datapump oder gar Export/Import. Dieses Feature lässt sich natürlich auch nur dann vernünftig nutzen, wenn man abhängige Objekte der Anwendung in einem oder wenigen Tablespaces hält.

Und wenn das Kind schon in den Brunnen gefallen ist? Dann gibt es einige Tools, die helfen können:

1. Offline
   Einfaches Verschieben der Tabelle: `ALTER TABLE <Tablelle> MOVE TABLESPACE <Tablespace>;`
   Oder bei mehreren Objekten vielleicht besser Datapump oder wenn man sich damit wohler fühlt und die Zeit hat Export/Import.
2. Online

Das Datenbank Redefinitionspaket: `DBMS_Redefinition`, wobei die Interim Table in einem anderen Tablespace angelegt wird (Vorsicht mit `LONG`, `BFILE` und `User Defined Types`!)
Alternativ gehen auch Third Party Lösungen zum Beispiel für das einfache Verschieben oder Reorganisieren eines Objektes den Space Manager with Live Reorg, oder, zum Synchronisieren verschiedener Objekte, Schemata oder Datenbanken, Shareplex

Eigenschaften von Tablespaces und Datendateien

Die Eigenschaften von Tablespaces und Datendateien betrachtet man am besten zusammen. Eine Datendatei ist wie eingangs beschrieben immer genau einem Tablespace zugeordnet.

TABLESPACE
Art des Tablespaces, der Default ist Permanent, es gibt aber auch `TEMPORARY` und `UNDO` Tablespaces.
`TEMPORARY` - Tablespace für Temporärsegmente, die für Sortierungen genutzt werden, wenn die Sortierung nicht über Indizes erfolgen oder im Speicher durchgeführt werden kann
`UNDO` - Tablespace für die vom System verwalteten UNDO Segmente

DATAFILE
Name der Datendatei, den kann aber auch als Oracle Managed File von der Datenbank vergeben lassen.

SIZE
Initialgröße der Datendatei, bei Oracle Managed Files standardmäßig 100 MB

SMALLFILE / BIGFILE (ab 10g)
`SMALLFILE` (default) - mehrere Datendateien möglich, 4M Blöcke, also bei 8K Blocksize: 4M * 8KB Blocksize = 32 GB
`BIGFILE` - nur eine große Datendatei (4G Blöcke, also bei 8K Blocksize (4 * 10^9) * (8 * 10^3) Byte = 32 * 10^12 Byte = 32 TB)
Die Größenänderung ist hier dann über Angabe des Tablespacenamen möglich, bei einem Smallfile Tablespace muss die Datendatei angegeben werden.

> Besonderheit: Außer bei SYSTEM, TEMP und UNDO Tablespaces sind Bigfiles nur mit Locally Managed Tablespaces und ASSM möglich. Ein Restore und Recovery einer Datendatei ist dann gleichbedeutend mit Restore und Recovery eines Tablespaces.

AUTOEXTEND
Gibt an, ob der Tablespace wachsen darf. Wenn man `AUTOEXTEND` verwendet, sollte man auch die `NEXT` und `MAXSIZE` Klauseln angeben.

BLOCKSIZE
Gibt an, wie groß die Blöcke sind. Ohne Angabe wird hier die Standardeinstellung verwendet, deren Default 8K ist. Die Blocksize hat Einfluß auf die maximale Größe eines Tablespaces und auch auf die Anzahl der IO Operationen, die durchgeführt werden müssen, um zum Beispiel an eine bestimmte Zeile zu kommen.

NOLOGGING / LOGGING / FORCE LOGGING
Tabellen können mit `NOLOGGING` angelegt werden, dann werden beim `CREATE TABLE (AS SELECT)` oder `bei INSERT /*+APPEND*/` keine Redo Log Informationen geschrieben. Das hat IO Performance Vorteile, es kann für die Tabellen aber kein Recovery durchgeführt werden. `LOGGING` und `NOLOGGING` gibt nur den Tablespace Default für das Anlegen von Tabellen an, mit `FORCE`

LOGGING auf Datenbank- oder Tablespaceebene wird ein Schreiben der Redologs für die entsprechenden Objekte erzwungen.

EXTENT MANAGEMENT LOCAL | DIICTIONARY
Die Standardeinstellung ist LOCAL und bewirkt, dass die Verwaltung der Extents im Tablespace Header stattfindet, statt wie bei DICTIONARY Managed Tablespaces im Data Dictionary, was zu Contention führt. Locally Managed Tablespaces haben noch einige andere Vorteile und sollten gewählt werden, Dictionary Managed Tablespaces sieht man meist als Altlasten, wenn Datenbanken mit dem Database Upgrade Assistant (DBUA) hochgezogen wurden.

SEGMENT SPACE MANAGEMENT
Gibt an, ob die Segmentspeicherverwaltung automatisch AUTO oder manuell MANUAL erfolgen soll. AUTO heißt, dass die Verwaltung der Füllgrade der Blöcke im Extent selbst über eine Bitmap verwaltet werden. Das macht es überflüssig, einen PCT_USED Wert anzugeben, oder sich um Freelists zu kümmern, das Segment verwaltet seinen Speicherplatz selbst. Das vermeidet Contention durch die Verwaltung im Data Dictionary.

Beachtenswerte Dinge – Best Practices

Es gibt einige Dinge, die man grundsätzlich beim Planen der Tablespaces und auch später beim Arbeiten mit der Datenbank beachten sollte.

- Der SYSTEM Tablespace gehört ausschließlich der Datenbank, User, Utilities und die Anwendung sollten dort keine Objekte anlegen. Vorsicht beim Arbeiten mit den Usern SYS oder SYSTEM, dann ist der SYSTEM Tablespace der Default Tablespace.
- Nicht zu viele Tablespaces anlegen, die dazu eindeutig benannt sein sollten, sonst verliert man leicht den Überblick. Wichtig ist dann noch sinnvolle Datendateiennamen, oder Oracle Managed Files zu verwenden.
- Die Namen Case insensitive und ohne Umlaute wählen, es ist zwar auch anders möglich, aber man muss ja nicht alles mitmachen. Es kann sonst sein, dass nicht jedes Feature funktioniert.
- Blocksize: Die Größe eines Blocks sollte immer gleich der Betriebssystem IO Größe sein, oder ein Vielfaches davon. So vermeidet man, dass um an einen Block zu kommen unnötig viele IO Operationen durchgeführt werden müssen, und dass bei einer IO Operation auch irrelevante Daten mitgelesen werden.
- Bei AUTOEXTEND immer eine NEXT Klausel setzen, sonst wird bei jeder Erweiterung immer nur ein Block allokiert (außer bei Oracle Managed Files, da sind es immer 100M). Dann auch eine MAXSIZE setzen und die Datendateien nicht unendlich wachsen lassen, das Dateisystem ist ja auch nicht unendlich groß.
- LOBs ohne ASSM: Es gab unter 10g einige Probleme mit LOBs und ASSM, unter 11g wird es sich zeigen müssen, ob ASSM mit LOBs gut funktioniert.
- Bei LOB Tablespaces die Blocksize möglichst klein wählen, weil ein LOB Segment immer in Chunks allokiert wird, die wiederum ein Vielfaches der Blocksize groß sein müssen. Sind die LOBs üblicherweise deutlich kleiner als die Chunks, die mindestens einen Block groß sind, geht viel Platz verloren.
- BACKUP SPECIAL I: Auch Offline Backups immer mit RMAN und im Mountmodus durchführen. Sonst weiß man nie, ob man alle Datafiles erwischt hat. Das böse Erwachen kommt dann, wenn doch mal ein Restore nötig wird…
- BACKUP SPECIAL II: Export eignet sich aufgrund der langen Import Zeit nicht als Backup, ein es ist auch nicht möglich die Datenbank nach dem Import weiter vorwärts zu rollen. Ein

Export ohne gesetzten `FLASHBACK_SCN` Parameter ist nicht konsistent! (Bei `CONSISTENT=Y` werden nur abhängige Objekte mit FK Constraints konsistent gespeichert.)

Fazit

Es lohnt sich, sich mit dem Thema Tablespaces noch weiter zu beschäftigen, schließlich geht es hier um eines der elementaren Dinge beim Arbeiten mit einer Datenbank, um deren Layout. Dieser Vortrag war natürlich nur ein grober Überblick über die Möglichkeiten die es gibt. Ein gutes Tablespacelayout kann einem DBA das Leben sehr einfach machen.

Kontaktadresse:

Thomas Klughardt
Quest Software GmbH
Im Mediapark, 4e
D-50670 Köln

Telefon: +49 (0) 221-5777 4114
Fax: +49 (0) 221-5777 452
E-Mail Thomas.Klughardt@quest.com
Internet: www.quest.com