Datenbanken II

Datenbankobjekte

von
Werner Hahn, 05IND-P

Inhaltsverzeichnis

1. Tabellen

In diesem Abschnitt soll auf die logische Speicherung von Datenbankobjekten am Beispiel von Tabellen eingegangen werden.

1.1. Relationale Tabellen

Relationale Tabellen sind der üblichste Typ in einer Datenbank, Daten zu speichern. Die Datensätze werden dabei über einen Heap organisiert, das heißt, die einzelnen Datensätze werden ohne besondere Berücksichtigung der Reihenfolge gespeichert. Beim Anlegen einer Tabelle kann auch besonders auf diese Eigenschaft hingewiesen werden:

```
CREATE TABLE tabellenname (
     Spaltendefinitionen,
     ...
)
ORGANIZATION HEAP;
```

Diese Bezeichnung dient jedoch allenfalls der Lesbarkeit, da dies sowieso die Standardeinstellung ist.

Jede Zeile einer relationalen Tabelle besteht aus einer oder mehreren Spalten; jede Spalte wird definiert durch einen Datentyp und eine Länge. Die folgende Tabelle geht beispielhaft auf einige der gebräuchlichsten in Oracle vordefinierten Typen ein:

Datentyp	Eigenschaft
VARCHAR2(n)	Eine Zeichenkette variabler Länge (max. 4000 Bytes). In bisherigen Oracle-Versionen ist dieser Befehl bedeutungsgleich mit VARCHAR(n), es wird allerdings im Hinblick auf Zukunftssicherheit empfohlen, immer VARCHAR2(n) zu verwenden.
NUMBER(p, s)	Ein numerischer Wert. Die Präzision p gibt dabei die Gesamtzahl der Stellen an, die Skalierung s die Zahl der Stellen vor (s negativ) oder nach dem Komma (s positiv).
DATE	Ein Datum.
CLOB	Ein *großes* Objekt aus Zeichen. In CLOBs können bis zu (4GB-1) * DB_BLOCK_SIZE Bytes gespeichert werden.
BLOB	Ähnlich einem CLOB, nur dass beliebige binäre Daten erlaubt sind.

Table 1: Datentypen in Oracle

Weiterhin können Spalten in einer relationalen Tabelle aus VARRAYs oder nested tables bestehen. Beide Spaltentypen sind eindimensional und können einem Datensatz in einer Spalte eine Menge von Werten zuordnen. Der Hauptunterschied zwischen diesen beiden Konstrukten besteht darin, dass VARRAYs zur Definitionszeit eine vorgegebene Länge haben und darüber hinaus nicht mehr wachsen können, während nested tables beliebige Größen annehmen können. Außerdem wird bei VARRAYs die Reihenfolge der Elemente beibehalten, während nested tables wie gewohnt über einen Heap organisiert werden.

1.2. Temporäre Tabellen

Der Name „temporäre Tabelle“ lässt vermuten, es handle sich um Tabellen, welche nicht dauerhaft abgespeichert werden. Dies ist jedoch irreführend: Nicht die Tabellen selbst, sondern die Daten sind temporärer Natur. Eine temporäre Tabelle kann mit dem SQL-Befehl `CREATE GLOBAL TEMPORARY TABLE` erstellt werden.
Ein Nutzer, der Rechte auf eine bestimmte temporäre Tabelle hat, kann entsprechend dieser Rechte Daten aus einer solchen Tabelle lesen, sowie DML-Operationen durchführen. Jeder Nutzer sieht hierbei jedoch nur seine eigenen Daten; die Daten der anderen Nutzer werden vom DBMS verborgen.
Temporäre Tabellen werden ihrerseits noch einmal in zwei Typen unterschieden: Im ersten Fall werden die Daten nur für die Dauer einer Transaktion gespeichert, im zweiten Fall für die Dauer einer Session. `ON COMMIT DELETE ROWS` löscht bei einem Commit oder Rollback alle Daten. `ON COMMIT PRESERVE ROWS` behält die Daten bei. Wird die Session durch den Nutzer beendet, werden die Daten jedoch auf jeden Fall gelöscht.
Für temporäre Tabellen werden keine Redo-Informationen abgespeichert.

1.3. Indexorganisierte Tabellen

Das Anlegen eines Indexes auf einer Spalte einer relationalen Tabelle kann in den meisten Fällen für einen stark beschleunigten Zugriff auf Datensätze dieser Tabelle über die entsprechende Spalte sorgen. Das DBMS legt hierzu als Index einen B-Baum an, welcher die Daten der indizierten Spalte noch einmal in einer geordneten Struktur ablegt (im

Gegensatz zum ungeordneten Heap). Stellt man jedoch fest, dass man in einer Tabelle ausschließlich über eine ganz bestimmte Spalte nach Datensätzen sucht, bietet es sich unter Umständen an, auf indexorganisierte Tabellen zurückzugreifen.
Bei einer indexorganisierten Tabelle wird auf den separaten Index zur Tabelle verzichtet und die kompletten Datensätze werden im Index abgelegt.

1.4. Object Tables

Object Tables werden seit Oracle8 durch Oracle unterstützt, um die Trennung zwischen Datenbank und Programmierung in Zeiten fortschreitender objektorientierter Programmierung möglichst gering zu halten. Die Objekte eines Object Tables werden über eine Object ID (OID) eindeutig bestimmt, es ist jedoch auch möglich, zusätzlich noch einen eigenen Primary oder Unique Key zu bestimmen.
Das folgende Beispiel soll demonstrieren, wie Objekte in Oracle genutzt werden können:

```
CREATE TYPE auto_typ AS OBJECT (
     marke               VARCHAR(30),
     modell              VARCHAR(40),
     kilometerstand      NUMBER(8,1)
);

CREATE TABLE autos OF auto_typ;

INSERT INTO autos
     VALUES(auto_typ('VW', 'Golf', 12000));
```

Andere Tabellen können nun auch Referenzen auf dieses Objekt (Golf) referenzieren, ohne von der zugehörigen Tabelle (autos) Kenntnis zu haben.

1.5. Externe Tabellen

Externe Tabellen sind keine Tabellen im herkömmlichen Sinn. Es kann sich dabei z. B. um Textdateien handeln, welche im Dateisystem auf der Festplatte liegen. Externe Tabellen

werden in zwei Teilen eingebunden. Zum einen muss die Tabelle angelegt werden und zum anderen muss für das Format der externen Datei ein Mapping zu Spalten und Zeilen für die Datenbank erstellt werden.
Wurde die Tabelle angelegt, erzeugt Oracle die entsprechenden Metadaten für den Oracle Data Dictionary, die Daten selbst werden jedoch nicht in die Datenbank übernommen, sondern verbleiben in der externen Datei.
Auf externe Tabellen können keine Indexe angelegt werden, weiterhin sind DML-Operationen nicht möglich, man ist also auf reine Queries beschränkt.

1.6. Geclusterte Tabellen

Geclusterte Tabellen unterscheiden sich stark von den namensgleichen geclusterten Indexen. Während ein geclusterter Index sicherstellt, dass die Daten in der Tabelle die gleiche Reihenfolge wie die Einträge im Index haben, eignen sich geclusterte Tabellen in Fällen, in denen zwei Tabellen häufig über die gleichen Spalten z. B. per Join verknüpft werden.
Hat man eine Tabelle mit Zeilen, die über einen Fremdschlüssel eine Zeile einer anderen Tabelle referenzieren (z. B. Detaildaten zu einer bestimmten Bestellung), so können die gemeinsamen Daten dieser beiden Tabellen im gleichen Block abgespeichert werden. Da die Datenbank blockweise von der Festplatte liest, verringern sich dadurch die I/O-Kosten, da nur noch ein Mal von der Platte gelesen werden muss, statt zwei Mal. Ein weiterer Vorteil ist, dass diese gemeinsamen Daten (auch Cluster Key Value genannt) nur einmal abgespeichert werden müssen und somit Platz sparen. Der Cluster Key Value wird außerdem in einem Cluster Index abgespeichert, was Zugriffe auf die Tabellen eines Clusters über den Cluster Key Value weiter beschleunigt.
Geclusterte Tabellen eigenen sich besonders für Tabellen, in denen verhältnismäßig viel gelesen, aber wenig verändert wird. Weiterhin schwindet der Vorteil geclusterter Tabellen, wenn häufig auf einzelne Tabellen eines Clusters zugegriffen wird, da man sich das Cluster dann sowieso hätte sparen können.

1.7. Hash Clusters

Hash Cluster ähneln den normalen geclusterten Tabellen. Der Zugriff auf die Cluster Key Values erfolgt hier jedoch nicht über einen Cluster Index, sondern über einen berechneten Hash. Der größte Performancevorteil von Hash Clustern bietet sich bei Vergleichsqueries, die einen exakten Wert abfragen, im Gegensatz zu Range Queries.

Eine Unterart der Hash Clusters sind die Sorted Hash Clusters. Hierbei ist es möglich, eine oder mehrere Spalten der Tabelle in aufsteigender Reihenfolge zu sortieren.

1.8. Partitionierte Tabellen

Partitionierte Tabellen werden eingesetzt, um sehr große Tabellen beherrschbarer zu machen. Für den Endanwender der Datenbank ist diese Partitionierung völlig transparent: Er muss nicht die richtige Partition auswählen, sondern betrachtet die Tabelle weiterhin als eine logische Einheit; das DBMS nimmt die Unterscheidung in Partitionen vor.
Partitionen bieten Vorteile bei der Wartung der Datenbank. Da Partitionen nicht immer gleichzeitig vorhanden sein müssen und auch auf verschiedenen physischen Datenträgern liegen können, kann im Falle eines Datenträgerschadens der Rest der Tabelle, welcher in den Partitionen auf den intakten Datenträgern liegt, weiter verwendet werden. Auch können Partitionen einzeln offline gesetzt werden, um z. B. Backups anzulegen, ohne die komplette Tabelle offline nehmen zu müssen.
Jeder Datensatz einer partitionierten Tabelle kann sich nur in genau einer Partition befinden.
Es gibt drei Möglichkeiten, Tabellen zu partitionieren:

Range Partitions
Hierbei werden die Datensätze nach Schlüsseln verteilt, die in bestimmte Bereiche fallen. Ein Anwendungsbeispiel wäre die Partitionierung von Bestelldaten nach Jahr.

List Partitions
Hier fallen die Schlüssel in eine vorgegebene Liste von möglichen Schlüsseln, die zur Partitionierung herangezogen werden. Sammelt man z. B. bundeslandbezogene Daten,

kann eine Partitionierung nach Bundesland erfolgen (von denen es bekanntlich nur 16, also eine feste Anzahl, gibt).

Hash Partitions

Können die Daten nicht wie bei den eben erwähnten Methoden in logische Partitionen unterteilt werden, bieten sich Hash Partitions an. Hierbei wird für jeden Datensatz ein Hash berechnet, der die Zuordnung in eine Partition vornimmt.

Des weiteren können partitionierte Tabellen auch in *Composite Partitions* unterteilt werden. Diese bestehen aus mehreren Schichten von Partitionierungen, z. B. erst nach Jahr (Range Partition) und dann nach Bundesland (List Partition).

2. Logische Speicherstrukturen in Oracle

Oracle verwendet eine Reihe von Unterteilungen, wenn es um die logische Speicherung von Datenbankobjekten geht. Die höchste Abstraktionsebene ist der Tablespace, welcher in Segmente unterteilt wird, die wiederum aus Extents bestehen. Die kleinste Einheit ist der Block.

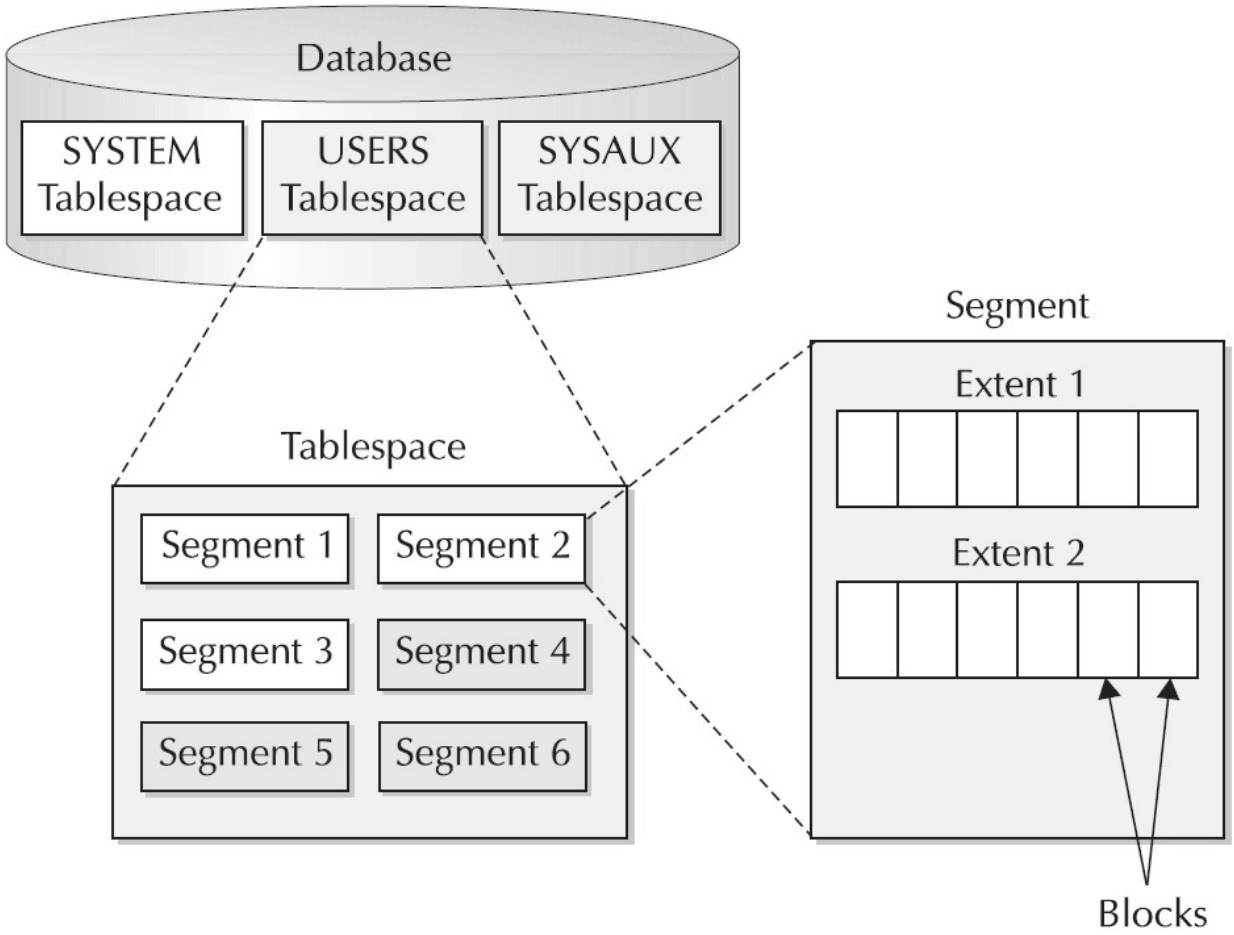

Illustration 1: Veranschaulichung logischer Speicherstrukturen

2.1. Tablespaces

Der Tablespace ist die größte Einheit in einer Datenbank, die zur Verwaltung der Daten dient. Eine Installation von Oracle 11*g* setzt zwingend zwei besondere Tablespaces voraus: SYSTEM und SYSAUX. Es handelt sich hierbei jedoch um das absolute Minimum und es wird dringend empfohlen, weitere Tablespaces einzurichten, damit das DBMS den Speicher, der ihm zur Verfügung steht, effektiver nutzen kann. Dies drückt sich auch

dadurch aus, dass eine Standardinstallation sechs Tablespaces anlegt. Beispielhaft sei der temporäre Tablespace erwähnt, welcher temporären Speicherplatz für z. B. Sortieroperationen bereitstellt, welche nicht in den Hauptspeicher des Servers passen.

2.2. Segmente

Ein Tablespace besteht aus einer Anzahl von Segmenten. Ein Segment wird vom DBMS insofern als Einheit behandelt, als dass sich in einem Segment immer nur ein Objekt befindet, z. B. eine Tabelle oder ein Index. Aufgrund dieser Aufteilung ist das Segment typischerweise auch die kleinste Speichereinheit, mit der ein Endanwender der Datenbanken zu tun hat.
In einer Oracle Datenbank wird zwischen vier verschiedenen Segmenten unterschieden:

2.2.1. Datensegmente

Jede Tabelle einer Datenbank wird in einem eigenen Datensegment abgespeichert. Handelt es sich bei der Tabelle um eine geclusterte oder partitionierte Tabelle, wird entsprechend der Anzahl der Cluster respektive Partitionen auch eine entsprechende Anzahl an Datensegmenten verwendet.

2.2.2. Indexsegmente

Indexsegmente speichern jeweils einen Index ab. Ähnlich wie bei Datensegmenten werden bei partitionierten Indexen entsprechend der Anzahl der Partitionen mehrere Indexsegmente verwendet.

2.2.3. Temporäre Segmente

Temporäre Segmente werden für Operationen verwendet, die mehr Speicher benötigen, als noch in den Hauptspeicher passt, z. B. Sortieroperationen. Temporäre Segmente werden nur für ein einzelnes SQL-Statement angelegt und verwendet.

2.2.4. Rollbacksegmente

Rollbacksegmente werden zur Speicherung der alten Daten bei DML-Operationen verwendet. Dadurch ist es möglich, bei einem manuellen Rollback oder auch bei einem Systemabsturz den alten Zustand der Datenbank wiederherzustellen. Außerdem können

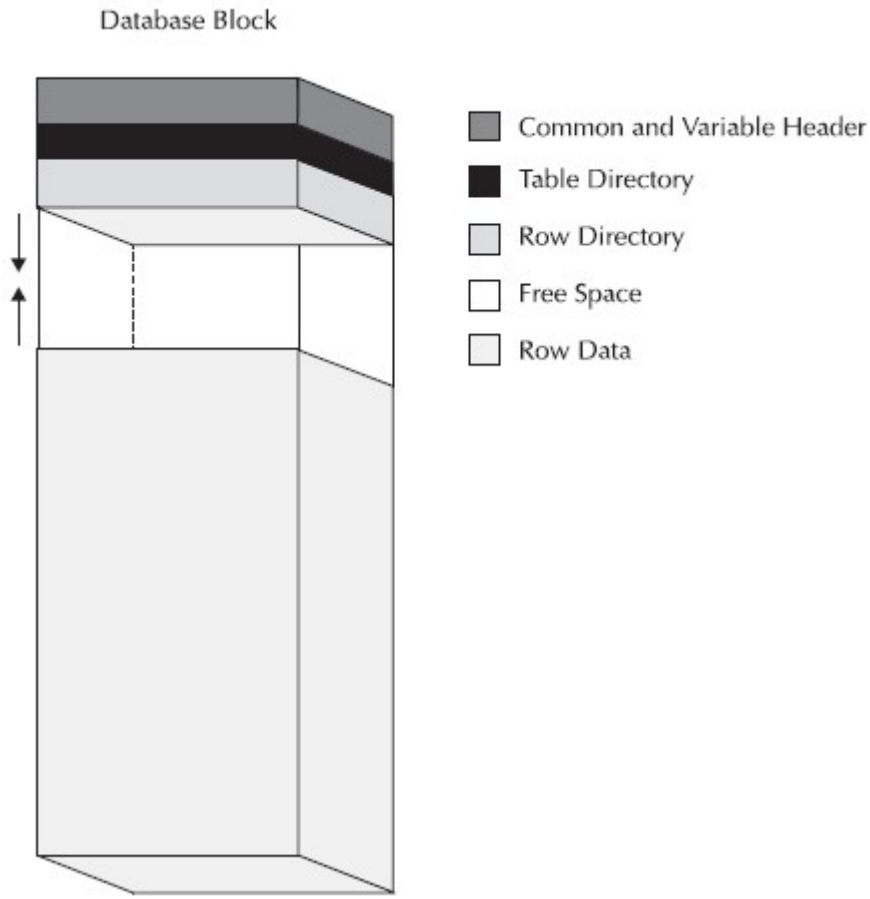

Illustration 2: Struktur eines Blockes

so andere Nutzer auf konsistente Daten der Tabelle zurückgreifen, auch wenn diese gerade geändert wird.

Seit Oracle 9*i* unterstützt die Datenbank das Automatic Undo Management, welches eine automatische Verwaltung von Rollbacksegmenten bzw. Undosegmenten vornimmt. In früheren Versionen war dies ausschließlich Sache des DBA. Inzwischen geht man bei Oracle jedoch davon aus, dass das Automatic Undo Management diese Verwaltung viel effizienter als der DBA vornehmen kann. Dies drückt sich auch in der Behandlung der manuellen Verwaltung aus: In Oracle 10*g* wurde sie missbilligt, in Oracle 11*g* wird standardmäßig das Automatic Undo Management verwendet und in zukünftigen Versionen soll eine manuelle Handhabung nicht mehr möglich sein.

2.3. Extents

Extents sind nicht nur in Datenbanken zu finden: Bei einem Extent handelt es sich generell um ein Stück zusammenhängenden Speichers, z. B. auch in Dateisystemen. Viele Extents

ergeben zusammen ein Segment. Wird ein Datenbankobjekt vergrößert, so wird zusätzlich benötigter Speicher immer als Extent allokiert und nicht z. B. blockweise.
Wird ein neues Datenbankobjekt erstellt, so wird initial ein Extent allokiert. Ist dieser Speicherplatz aufgebraucht, werden neue Extents entsprechend einer eingestellten Strategie allokiert. Diese wird vom Tablespace vorgegeben, kann jedoch bei der Erstellung einer Tabelle überschrieben werden. Dabei gibt das Schlüsselwort UNIFORM an, dass weitere Extents eine konstante Größe besitzen sollen. Variable Größen neuer Extents erhält man mittels AUTOALLOCATE. Hierbei wird das erste neue Extent 64KB groß und alle weiteren werden aus einem Vielfachen des initialen Extents errechnet.
Extents einer Tabelle werden niemals gelöscht, auch wenn alle Datensätze einer Tabelle gelöscht werden. Erst wenn die Tabelle gedropt oder „verkürzt“ wird, werden Extents auch gelöscht. Die maximale Anzahl von Extents, die eine Tabelle besitzen darf, wird als High Water Mark (HWM) bezeichent.

2.4. Blöcke

Ein Block ist die kleinste Speichereinheit in einer Oracle Datenbank. Die Größe eines Blockes sollte immer ein ganzzahliges Vielfaches der Blockgröße des Betriebssystems sein, damit I/O-Operation effektiv durchgeführt werden können. Die Standardgröße eines Blockes wird im Oracle Parameter `DB_BLOCK_SIZE` abgelegt. Die Tablespaces SYSTEM und SYSAUX müssen diese Größe auch tatsächlich verwenden. Es ist allerdings noch möglich, vier weitere Blockgrößen zu definieren, die von anderen Tablespaces verwendet werden können.
Die Struktur eines Blocks ist in Illustration 2 dargestellt. Der Header eines Blocks gibt Rückschlüsse auf die Art der Daten, die sich in ihm befinden: Tabellen oder Indexe. Das Table Directory enthält Einträge für alle Tabellen, welche Daten in diesem Block speichern. Meistens ist dies nur eine, da ein Segment nur eine Tabelle oder einen Index enthalten kann, im Falle von geclusterten Tabellen können jedoch mehrere Tabellen Daten in einem Block abspeichern. Das Row Directory enthält Einträge über alle Tabellenzeilen, die im Block abgespeichert sind.
Diese Metadaten machen nur einen kleinen Prozentsatz der eigentlich in einem Block zu speichernden Daten aus. Der Großteil ist für die eigentlichen Datensätze vorgesehen. Doch auch dem freien Platz wird eine Bedeutung zugemessen: Jeder Block wird nur bis zu

einem bestimmten Prozentsatz gefüllt. Dieser Prozentsatz wird zur Erstellungszeit des Blocks durch den Parameter PCTFREE bestimmt. Sind mehr Daten in einem Block als PCTFREE angibt, dürfen in den Block keine weiteren Daten eingefügt werden. Dieser Platz wird benötigt, falls einzelne Spalten einer Zeile durch UPDATE-Statements an Größe gewinnen. Es muss also nicht sofort ein neuer freier Block gesucht werden, sondern der freie Platz kann effektiv genutzt werden.

Passt ein einzelner Datensatz nicht in einen Block, weil er größer als die Blockgröße ist, wird er über mehrere möglichst zusammenhängende Blöcke verteilt. Wird ein Datensatz jedoch durch UPDATEs zu groß für einen Block, so *migriert* Oracle diesen Datensatz: Die komplette Zeile wird in einen neuen Block verschoben und an der alten Position wird ein Pointer auf die neue Position gesetzt. Selbstverständlich nimmt die I/O-Performance stark ab, je mehr Pointer in einem Block existieren, da sich dadurch die Zugriffskosten verdoppeln können. In diesem Fall muss entweder der Parameter PCTFREE angepasst oder die Tabelle neu erstellt werden. Das Tuning und die Wartung von Datenbanken ist jedoch nicht Bestandteil dieser Ausarbeitung.

Seit Oracle 9*i* bietet Oracle das Automatic Segment Space Management (ASSM) an, welches den freien Speicherplatz in Segmenten effizient verwalten soll. Dabei wird in jedem Segment ein Bitmap Block angelegt, welcher eine Übersicht über den freien Speicher der Blöcke gibt. Diese Information wird wie folgt kodiert:

Bitmap	Beschreibung
0000	unformatierter Block
0001	voller Block
0010	weniger als 25% freier Platz
0011	25% bis 50% freier Platz
0100	50% bis 75% freier Platz
0101	Mehr als 75% freier Platz

Table 2: Bitmap für die Gefülltheit von Blöcken

In einem Oracle Real Application Server wird das Speichermanagement nur noch über diese Bitmaps durchgeführt. Eine Verwendung so genannter Freelists, welche im Tablespace oder dem Index Header abgespeichert werden müssen, entfällt somit. Weiterhin werden Parameter wie PCTFREE ignoriert, da sie für ASSM nicht von Nöten sind.

3. Oracle Data Dictionary

Der Oracle Data Dictionary ist eine Sammlung von Metadaten über die Datenbank. Er kann über verschiedene Tabellen und Views eingesehen werden.

3.1. Metadaten über Metadaten

Der Oracle Data Dictionary beinhaltet nicht nur Metadaten über die eigentlichen in der Datenbank abgespeicherten Daten, sondern auch Metadaten über sich selbst. Dazu gibt es die beiden Views DICTIONARY und DICT_COLUMNS. Die Tabelle DICTIONARY enthält eine Beschreibung jeder View des Data Dictionary.
Möchte man z. B. wissen, auf welche Informationen über Tabellen man Zugriff hat, kann man folgendes SQL-Statement benutzen:

```
SELECT *
FROM DICTIONARY
WHERE table_name LIKE 'USER%' AND table_name LIKE '%TABLE%';
```

Ein Ausschnitt der Antwort sieht wie folgt aus:

TABLE_NAME	COMMENTS
USER_ADVISOR_SQLW_TABLES	-
USER_ALL_TABLES	Description of all object and relational tables owned by the user's
USER_BASE_TABLE_MVIEWS	All materialized views with log(s) owned by the user in the database
USER_EVALUATION_CONTEXT_TABLES	tables in user rule evaluation contexts
USER_EXTERNAL_TABLES	Description of the user's own external tables

Table 3: auszugsweise Ausgabe aus der View DICTIONARY

Interessiert man sich für die Spalten einer bestimmten View, so kann die View DICT_COLUMNS zu Rate gezogen werden:

```
SELECT *
FROM DICT_COLUMNS
WHERE table_name = 'USER_ALL_TABLES';
```

Auch hier soll ein Ausschnitt aus dem 55 Zeilen umfassenden Resultat genügen (die Spalte TABLE_NAME wurde hier ausgelassen, da sie überall den gleichen Inhalt USER_ALL_TABLES hat):

COLUMN_NAME	COMMENTS
MONITORING	Should we keep track of the amount of modification?
CLUSTER_OWNER	Owner of the cluster, if any, to which the table belongs
DEPENDENCIES	Should we keep track of row level dependencies?
COMPRESSION	Whether table compression is enabled or not
DROPPED	Whether table is dropped and is in Recycle Bin

Table 4: auszugsweise Ausgabe aus der View DICT_COLUMNS

Diese beiden Views dienen mit ihren COMMENTS-Spalten als kleines Nachschlagewerk und Wegweiser durch den Oracle Data Dictionary.

3.2. Zugriffsbeschränkungen des Data Dictionary

Mit einer Datenbank arbeiten zumeist mehr als eine einzelne Person. Jede dieser Personen kann dabei Besitzer bestimmter Objekte sein. Das DBMS verhindert, dass ein Nutzer ohne ausreichende Rechte die Daten eines anderen Nutzers einsehen kann. Die Sichten des Oracle Data Dictionary sind auch entsprechend der Berechtigungen des anfragenden Nutzers gegliedert:
Das Präfix USER gibt an, dass die entsprechende View nur Metadaten über Objekte anzeigt, deren Besitzer der Nutzer ist, der die Anfrage stellt. Views mit dem Präfix ALL zeigen zusätzlich noch die Daten für alle Objekte, für die der aktuelle Nuter Leserechte besitzt. Das Präfix DBA ist dem Datenbankadministrator vorenthalten, welcher Metadaten von allen Objekten ungeachtet deren Besitzern ansehen darf.

3.3. Dynamic Performance Views

Eine Besonderheit des Oracle Data Dictionary stellen die Dynamic Performance Views dar. Mit ihnen lassen sich in Echtzeit Daten über die Performance der Oracle Datenbank erfragen. Die Views basieren auf Tabellen, welche alle mit V_$ beginnen. Diese Tabellen gehören dem besonderen Datenbankbenutzer SYS. Auf diese Tabellen wurden Views definiert und diese wiederum mit Synonymen versehen, welche mit V$ beginnen. Von Seiten Oracles wird empfohlen, wenn, dann nur über V$ Views auf die Performancedaten zuzugreifen, noch besser allerdings nicht direkt, sondern über den Oracle Enterprise Manager.
Als weiteren Zusatz existieren noch die GV$ Views. In geclusterten Datenbanken werden damit Informationen über alle laufenden Instanzen gesammelt, nicht nur über die aktuelle, wie durch die V$ Views. Die GV$ Views sind den V$ Views strukturell gleich, mit der Ausnahme, dass jede View noch um die Spalte INST_ID erweitert wird, welche die ID der entsprechenden Instanz angibt.
Ein Beispiel für eine Performance View ist V$SYSSTAT, welche Systemstatistiken auflistet:

Column	Description
STATISTIC#	Statistic number
NAME	Statistic name
CLASS	A number representing one or more statistics class.
VALUE	Statistic value
STAT_ID	Identifier of the statistic

Table 5: Spalten von V$SYSSTAT

Die Dynamic Performance Views sind von Version zu Version der Oracle Datenbank Schwankungen unterlegen. So weist Oracle im angegebenen Beispiel bei der Spalte STATISTIC# bereits drauf hin, dass diese Nummer nicht verlässlich ist und sich in jedem Release ändern kann.

4. Quellen

- B. Bryla, K. Loney: Oracle Database 11*g* DBA Handbook (2008), Oracle Press
- K. Loney: Oracle Database 10*g*: The Complete Reference (2004), Oracle Press
- J. Ahrends et al.: Oracle 10*g* für den DBA, Addison-Wesley (kostenlose Leseprobe)
- Oracle Database Online Documentation 10*g* Release 2, insbesondere http://download.oracle.com/docs/cd/B19306_01/server.102/b14220/datadict.htm und verlinkte