PEARSON Education

ADDISON-WESLEY

Johannes Ahrends, Dierk Lenz,
Patrick Schwanke, Günter Unbescheid

Oracle 10g für den DBA

Effizient konfigurieren, optimieren und verwalten

ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City

3 Datenbank-Architektur

3.1 Namenskonventionen

Beim Umgang mit einer Datenbank werden auf den verschiedensten Ebenen Dinge benannt: Datenbanken, Instanzen, Dienste, Dateien, Tablespaces usw. Neben den allgemein gültigen Regeln für die Namensvergabe, die z.B. die maximale Länge und die erlaubten Zeichen festlegen und vom Datenbanksystem vorgegeben werden, ist es empfehlenswert, eigene Richtlinien für die Namensbildung zu schaffen.

Warum denken wir an dieser Stelle über diese Namenskonventionen nach?

Einige der in Frage kommenden Beweggründe sind:

1. Wenn wir darüber nachdenken, nachdem die Datenbank konfiguriert und die Daten geladen sind, ist es zu spät.
2. Eine Oracle-Datenbank wird auf verschiedenen Ebenen mit diversen Namen identifiziert (u.a. Datenbankname, Instanzname, Net-Aliasname sowie Dienstname). Sie können alle verschieden sein, müssen aber nicht.
3. Es gibt immer mehr Situationen, bei denen die im Unternehmen vorhandenen Datenbanken global betrachtet werden – der Enterprise Manager oder ein Storage-Subsystem bzw. SAN sind nur zwei Beispiele. Wenn dann fünf von acht Datenbanken `ORACLE` heißen, ist die Übersichtlichkeit dahin.

Im Folgenden wird zunächst über Datenbanken und deren Namensgebung diskutiert; im Anschluss daran über die Namensgebung bei Systemobjekten, also Tablespaces und Datendateien.

Groß-, Kleinschreibung und Sonderzeichen

Es ist möglich, alle Benennungen sowohl in Groß- als auch in Kleinbuchstaben vorzunehmen. In vielen Teilen sind außerdem auch Sonderzeichen möglich. Man muss allerdings zwischen der einfachen Schreibweise (`CREATE TABLESPACE temp`) und der tatsächlichen Maskierung (`CREATE TABLESPACE "temp"`) unterscheiden. Im zweiten Fall wird tatsächlich ein Data-Dictionary-Eintrag mit Kleinbuchstaben generiert, während im ersten Fall Großbuchstaben verwendet werden.

In der Praxis sollten Sie allerdings beachten, wenn Sie tatsächlich Sonderzeichen und Kleinbuchstaben verwenden, dass einige Tools unter Umständen nicht mehr funktionieren, da nicht jedes Werkzeug auf diese Besonderheit ausgelegt ist. Außerdem müssen Sie bei Abfragen ebenfalls diese Maskierung benutzen.

In der Regel werden wir daher unmaskierte Namen verwenden, so dass wir ohne Maskierung auskommen. Für die ORACLE_SID verwenden wir außerdem oftmals auf Betriebssystemebene Großbuchstaben, da diese dadurch bei Pfadangaben oder Dateinamen besser erkennbar ist.

3.1.1 Datenbanken

Bei der Benennung von Datenbanken ist zunächst zu unterscheiden zwischen dem per Parameter vergebenen Datenbanknamen (`db_name`), dem Domänennamen der Datenbank (`db_domain`) und dem *globalen* Datenbanknamen, der aus den beiden genannten Parametern zusammengesetzt wird, jedoch auch – unabhängig davon – explizit gesetzt werden kann. Die Instanz, also die Prozess- und Speicherstrukturen, die auf die Datenbank zugreifen, werden über einen Instanznamen (`ORACLE_SID`) identifiziert. Die folgenden Abschnitte erläutern die Zusammenhänge zwischen diesen und weiteren Namen.

Domänen

Bei der Vergabe von Namen für eine Oracle-Datenbank gibt es zwei Stellen, an denen eine Domäne auftauchen kann:

- Die `db_domain` wird zusammen mit `db_name` (beides Serverparameter) interner Bestandteil des Datenbanknamens; zusammen ergeben diese Namen auch den Standardwert für den Serverparameter `service_names`, der ab Oracle8i zur Adressauflösung bei den Netzwerkdiensten genutzt werden kann.[1]
- Net-Aliasnamen, früher auch mit SQL*Net Alias oder Net8-Name bezeichnet, können ebenfalls mit einer Domäne versehen werden. Einen hohen Bekanntheitsgrad unter langjährigen Oracle-DBAs dürfte die Standarddomäne früherer Releases, `world`, haben.

Die Empfehlung lautet, die Domänen in beiden Fällen gezielt zu nutzen und identisch zu halten.

Mittels eines weiteren Serverparameters können die Domänen zusammen mit Datenbanknamen auf Datenbank-Links übertragen werden: Sobald

```
global_names = true
```

steht, dürfen Datenbank-Links nur so benannt werden wie der globale Name der Zieldatenbank. Diese Forderung ist gerade in *gewachsenen* Datenbanken schwer einzuhalten, da der funktionale Verlust, der durch das Deaktivieren eines Datenbank-Links entsteht, oft schwer zu überblicken ist. Fängt man jedoch mit einem neuen System *bei null* an, so ist diese Einstellung unbedingt zu empfehlen.

Einfache Empfehlungen für Standarddomänen sind `firma.de`, `firma.com` oder einfach `firma`. Da die im Oracle-Namensmodell verwendeten Domänen nicht identisch zu den (meist jedoch zumindest ähnlichen) Netzwerkdomänen sein müssen, können mit Hilfe der Domänen zusätzliche Informationen hinterlegt werden; z.B. können alle produktiven Datenbanken in die Domäne `prod.firma.de` gelegt werden.

1 In der `CONNECT_DATA`-Klausel wird hierzu statt des Parameters (`SID=XXX`) der Parameter (`SERVICE_NAME=XXX[.DOM]`) angegeben. Eine Verbindung wird hergestellt, wenn der angegebene `SERVICE_NAME` einem der in `service_names` angegebenen Namen entspricht.

Datenbanknamen

Ausgehend davon, dass Domänen verwendet werden, kann sich der Datenbankname auf die Funktion der Datenbank fokussieren. Datenbanknamen wie `oradb` sind wenig sinnvoll. Es sollten eher Namen wie `dwh` für Data Warehouse, `crm` für die Datenbank eines CRM-Systems oder `xyz` für die Datenbank unter der Anwendung XYZ ausgewählt werden.

Der Datenbankname kann insgesamt acht Zeichen lang sein. Im Hinblick auf die nächste Migration kann es sinnvoll sein, dem Datenbanknamen eine Nummer oder Versionsbezeichnung hinzuzufügen, also z.B. `pps01` oder `pps10g`.

Bei Datenbanken, die als RAC (Real Application Clusters, siehe Kapitel 13) ausgebaut werden, sollte der Datenbankname nicht länger als sieben Zeichen sein, um bei den Instanznamen (die ebenfalls maximal acht Zeichen lang sein dürfen) die Instanznummer als Suffix an den Datenbanknamen anhängen zu können.

Der Datenbankname wird als Serverparameter `db_name` hinterlegt. Er kann beim `CREATE DATABASE`-Kommando angegeben werden. Es wird grundsätzlich empfohlen, dies zu tun – damit hat man die Sicherheit, mit den richtigen Serverparametern zu arbeiten. Arbeitet man z.B. mit der falschen Serverparameterdatei, so geht das `CREATE DATABASE`-Kommando schief, und das ist gut so.

Der *globale Datenbankname* wird beim Anlegen der Datenbank aus den Serverparametern `db_name` und `db_domain` abgeleitet, und zwar als `db_name.` `db_domain`. Dieser kann über die View `global_name` abgefragt werden:

```
SQL> SELECT global_name FROM global_name;
```

Er kann nach Erstellung der Datenbank unabhängig vom eigentlichen Datenbanknamen und der Domäne beliebig geändert werden:

```
SQL> ALTER DATABASE RENAME GLOBAL_NAME TO globdb.firma.de;
```

Der globale Datenbankname wird dazu verwendet, die Namen von Datenbank-Links zu erzwingen. Mit gesetztem Serverparameter

```
global_names = true
```

muss – wie bereits erwähnt – ein Datenbank-Link immer genauso heißen wie der globale Datenbankname der Zieldatenbank.

Eine nachträgliche Änderung des Datenbanknamens kann über das Kommandozeilen-Tool `nid` erfolgen. Dazu muss die Datenbank exklusiv gemountet sein, also sich im MOUNT-Zustand befinden, und (im Falle einer RAC-Datenbank) der Serverparameter `cluster_database=FALSE` gesetzt sein. Bei gesetzter Umgebungsvariable `ORACLE_SID` ändert folgender Aufruf dann den Datenbanknamen:

```
$ nid target=sys/passwort dbname=newname
```

Anschließend muss die Datenbank heruntergefahren und der Serverparameter `db_name` entsprechend angepasst werden. Bei Oracle 10g geht dies über ein Kommando der Form:

```
SQL> ALTER SYSTEM SET db_name = newdb SCOPE=SPFILE;
```

Anschließend kann die Datenbank wieder hochgefahren werden.

Eindeutigkeit von Datenbanknamen

Vor der Version Oracle 10g galt die Restriktion, dass unterschiedliche Datenbanken auf demselben Knoten unterschiedliche Datenbanknamen (db_name) haben müssen. In den meisten Fällen ist dies nicht weiter problematisch. Manchmal kommt es allerdings vor, dass man zu Testzwecken eine Datenbank auf demselben Knoten klonen möchte. In diesem Fall musste man zunächst die Referenzdatenbank herunterfahren, die Klondatenbank mounten, mit dem oben beschriebenen nid-Werkzeug den db_name ändern, um anschließend beide Datenbanken hochfahren zu können.

Dieses etwas umständliche Verfahren ist mit Oracle 10g nicht mehr notwendig. Möglich macht dies der neue Serverparameter db_unique_name. Unterschiedliche Oracle 10g-Datenbanken auf demselben Knoten müssen lediglich unterschiedliche Werte für db_unique_name haben. Der db_unique_name lässt sich einfach über die Parameterdatei oder im NOMOUNT-Status der Datenbank ändern:

```
SQL> ALTER SYSTEM SET db_unique_name = neudb SCOPE=SPFILE;
```

Die db_unique_names von 10g-Datenbanken sowie die db_names älterer Datenbanken müssen nun also pro Knoten eindeutig sein. Der db_unique_name ist – wenn nicht explizit gesetzt – mit dem db_name identisch.

Für Oracle 10g-Datenbanken besteht daher eigentlich kein Grund mehr, den db_name nachträglich zu ändern. Es sollte – wenn nichts dagegen spricht – auf das einfachere Verfahren der Änderung des db_unique_name ausgewichen werden.

Achtung:
Verstößt man versehentlich gegen die genannten Eindeutigkeitsforderungen, lässt sich die zweite Instanz zwar noch hochfahren, den MOUNT-Versuch quittiert Oracle aber mit der etwas irritierenden Fehlermeldung:

```
ORA-01102: Datenbank kann nicht im EXCLUSIVE-Modus mit Mount angeschlossen werden.
```

Da der db_unique_name auch noch in anderen Zusammenhängen eine wichtige Rolle bei der Namensgebung spielt, beispielsweise für Data-Guard-Konfigurationen (siehe Kapitel 13) sowie für Oracle Managed Files (siehe Kapitel 3.9), ist es zu empfehlen, ihn nicht nur pro Knoten, sondern über die gesamte Datenbankumgebung eindeutig zu halten. Dies lässt sich u.U. am einfachsten erreichen, indem er mit dem globalen Datenbanknamen gleichgesetzt wird. Die Länge des db_unique_name ist allerdings auf 30 Zeichen begrenzt.

Instanznamen

In der Oracle-Architektur sind Datenbank und Instanz getrennt, und somit können auch unterschiedliche Namen vergeben werden. Der Instanzname verfügt nicht über eine Domäne; er muss allerdings – wie der Datenbankname auch – pro Knoten eindeutig sein.

Auch wenn es nicht zwingend erforderlich ist – der Instanzname sollte gleich oder zumindest ähnlich dem Datenbanknamen sein. Auf Grund der fehlenden Domänen kann er aus Übersichtlichkeitsgründen leicht abgewandelt werden, z.B. kann die Instanz zur Datenbank dwh10g.prod.firma.de den Namen pdwh10g tragen, um die Eigenschaft Produktivsystem auch im Instanznamen zu verdeutlichen.

Die maximale Länge des Instanznamens, hinterlegt im Umgebungsparameter `ORACLE_SID`, ist plattformspezifisch. Acht Stellen werden bei den getesteten Plattformen problemlos akzeptiert; mehr als acht Stellen werden generell nicht empfohlen.

Bei RAC-Datenbanken empfiehlt es sich, die Instanzen mittels eines Suffix durchzunummerieren, z.B. `pdwh10g1` und `pdwh10g2`.

Datenbanken im Netzwerk

Sobald man auf eine Datenbank im Netzwerk zugreifen will, kommt zusätzlich zu den bereits diskutierten Namen ein Net-Aliasname ins Spiel. Hiermit wird die Datenbank im Netzwerk identifiziert.

Net-Aliasnamen können mit Domänen versehen werden. Sie sehen daher den globalen Datenbanknamen sehr ähnlich und werden oft mit diesen verwechselt.

Hat man sich bei den Datenbanknamen Gedanken gemacht, so kann man natürlich die Net-Aliasnamen identisch zu den Datenbanknamen inklusive Domäne wählen.

Oft gibt es für eine Datenbank mehrere Net-Aliasnamen. Dabei wird für jede Anwendung ein spezieller Net-Aliasname verwendet, so dass die Anwendungsdaten bei Bedarf zwischen verschiedenen Datenbanken verschoben werden können.

Dieses Konzept findet sich auch bei den Dienstnamen als Serverparameter wieder: Hiermit kann jeder Instanz eine Liste von Diensten (Serverparameter `service_names`) mitgegeben werden. Der Parameter wird standardmäßig auf den Wert `db_name.db_domain` gesetzt. Beim Verbindungsaufbau über die Oracle Net-Schnittstelle wird ein `service_name` mitgegeben, der mindestens einem Wert aus der Liste der `service_names` der Instanz entsprechen muss.

3.1.2 Systemobjekte

Auch für Systemobjekte wie Tablespaces und Datendateien müssen Namen gefunden werden. Hierbei ist auf jeden Fall Kreativität gefragt, denn Systemobjekte sind für die Anwendung nicht unmittelbar erforderlich und spielen somit auch keine Rolle im System, die a priori benannt werden kann.

Daher ist es wichtig, sich einige Grundregeln zu überlegen, um den Überblick im Datenbankdschungel nicht zu verlieren.

Tablespaces für Systemobjekte wie Undo- und Temporärsegmente sollten immer gleich genannt werden, z.B. `undotbs` und `temp`. Bei Tablespaces für Anwendungsdaten bringt die Anwendung oft einen Vorschlag mit. Ansonsten nimmt man für eine Anwendung X gerne `x_data` für Tabellen, `x_index` für Indizes usw.

Folgt man der zumindest für sehr große Datenbanken empfehlenswerten Strategie, Tabellen und Indizes sehr unterschiedlicher Größen in unterschiedlichen Tablespaces zu speichern, sollte sich dies auch im Tablespace-Namen widerspiegeln, z.B. `x_data_big` für die größten Tabellen, `x_index_small` für Indizes auf kleinen Tabellen etc.

Bei den Datendateien sollte gewährleistet sein, dass man am voll qualifizierten Dateinamen folgende Eigenschaften erkennen kann:

- Diese Datei gehört zu einer Oracle-Datenbank (z.B. Pfad `/oradata`).
- Den Datenbanknamen (ohne Domäne) (z.B. Pfad `/oradata/PROD10G/`)
- Den Tablespace-Namen (z.B. Dateiname: `temp01.dbf`)
- Eine laufende Nummer für die n-te Datei des Tablespaces

Die Benennung von Rollback-Segmenten sollte kein Thema mehr sein, da seit der Version Oracle9i die automatische Undo-Verwaltung zu empfehlen ist (siehe Kapitel 3.3.5).

3.2 Komponenten einer Oracle-Datenbank

Auf physikalischer Ebene besteht eine Oracle-Datenbank aus einer Reihe von Komponenten. Im Standardfall sind diese Komponenten als Dateien innerhalb eines normalen Dateisystems (z.B. NTFS, ReiserFS etc.) ausgeführt. Es gibt aber auch die Möglichkeit, anstelle eines Dateisystems so genannte Raw-Devices zu benutzen oder – seit Oracle 10g – den mitgelieferten Volume-Manager ASM. Diese Methoden werden weiter unten besprochen. Der Einfachheit halber spricht man aber in allen Fällen von Dateien, den so genannten Datenbankdateien.

Folgende Komponenten sind unverzichtbar, gehören also in jedem Fall zu einer Oracle-Datenbank:

- *Datendateien* nehmen die eigentlichen Nutzdaten sowie sämtliche Metadaten auf. Sie determinieren in vielen Fällen den Gesamtspeicherbedarf einer Datenbank (eine Ausnahme bilden Datenbank mit sehr hoher Datenänderungsrate).
- *Online-Redolog-Dateien* bilden das Transaktionsprotokoll einer Datenbank. Es muss mindestens zwei Redolog-Dateien geben, typischerweise verwendet man zwischen drei und fünf (diese können ggf. noch gespiegelt werden, was die nominelle Anzahl entsprechend erhöht).
- Die *Kontrolldatei* enthält Metadaten über die physikalische Struktur und logische Konsistenz der Datenbank sowie backup-relevante Informationen. Sie ist typischerweise wenige MB groß und sollte einfach oder zweifach gespiegelt sein.
- Die *Parameterdatei* beinhaltet die gesamte Konfiguration der Oracle-Instanz. Sie ist nur wenige KB groß.

Einige weitere Komponenten sind optional:

- *Temporärdateien* sind Container für Daten, die nur temporär anfallen (z.B. Zwischenergebnisse für Sortierungen). Zwar kann eine Datenbank auch ohne Temporärdateien betrieben werden, in der Praxis sollte aber aus Performance-Gründen jede Datenbank mindestens einen Temporär-Tablespace aus Temporärdateien haben.
- *Archivierte Redolog-Dateien* sind – wie der Name andeutet – im Wesentlichen Archivierungen, also Kopien voll geschriebener Online-Redolog-Dateien. Für Datenbanken mit hoher Änderungsrate können diese einen erheblichen Anteil des Gesamtspeicherbedarf einer Datenbank ausmachen. Produktionsdatenbanken

sollten aus Gründen der Wiederherstellung im Fehlerfall unbedingt im Archivierungsmodus betrieben werden, für Test- und Entwicklungsumgebungen kann dies eventuell entfallen (siehe Kapitel 10.1).

- Die *Flashback-Log-Dateien* stellen eine Neuerung in Oracle 10g dar. Sie enthalten Abbilder alter Datenbankinhalte und werden für die Funktion `FLASHBACK DATABASE` benötigt. Je nach Änderungsrate der Datenbank und eingestellter Aufbewahrungszeit kann ein höherer Gesamtspeicherbedarf anfallen. Dieser kann aber durch geeignete Konfiguration nach oben limitiert werden. Weitere Informationen zu Flashback-Log-Dateien befinden sich in Kapitel 10.4.
- Die *Block Change Tracking-Datei* speichert Informationen darüber, welche Datenbankblöcke seit dem letzten Backup geändert wurden. Sie ist typischerweise wenige 10 MB groß.
- Die *Passwortdatei* erlaubt Administratoren eine spezielle Authentifizierung gegenüber der Datenbank, insbesondere um eine heruntergefahrene Datenbank zu starten (die normale Authentifizierung von Benutzern bei geöffneter Datenbank findet unabhängig davon durch die Datenbank selbst statt). Die Dateigröße beträgt wenige KB. Auf die Datei kann optional verzichtet werden, dann läuft die Authentifizierung über Gruppenzugehörigkeiten auf Betriebssystemebene.

Alle Komponenten werden in den folgenden Abschnitten dieses Kapitels näher beleuchtet.

3.3 Tablespaces und Datendateien

Alle Daten werden in der Oracle-Datenbank in Tablespaces verwaltet, die auf der physikalischen Ebene durch Datendateien (*Datafiles*) repräsentiert werden. Durch diese Unterscheidung zwischen logischer Architektur und physikalischer Speicherung ist es möglich, auf unterschiedlichen Plattformen (MS-Windows, Unix, OpenVMS, MVS etc.) den gleichen Aufbau der Datenbank zu gewährleisten.

Das folgende Bild macht den Zusammenhang zwischen physikalischen Platten, Datendateien und Tablespaces deutlich.

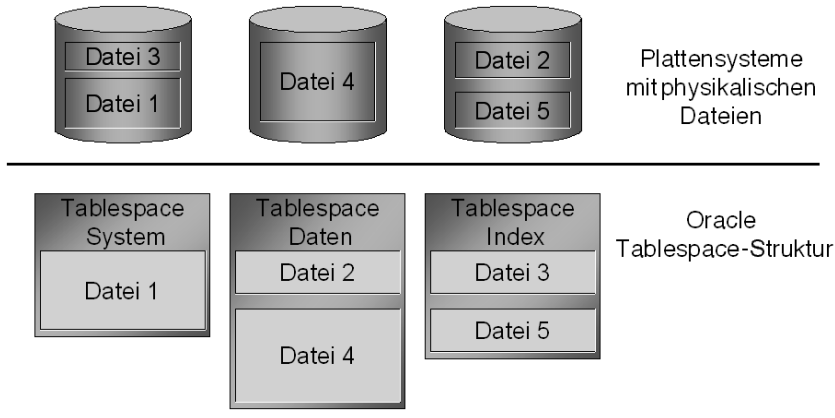

Abbildung 3.1: Tablespaces und Datendateien

3.3.1 Planung

Für die Planung einer Oracle-Datenbank stellt sich zunächst die Frage, wie man die Daten auf einzelne Tablespaces bzw. Datendateien verteilt. Dabei spielen unterschiedliche Gesichtspunkte eine Rolle:

- Performance
  Wie kann der Zugriff über mehrere IO-Kanäle optimiert werden, wie kann man Fragmentierungen vermeiden?
- Verfügbarkeit
  Wie kann im Fall eines Medienfehlers möglichst ohne Transaktionsverlust wieder aufgesetzt werden?
- Backup-Recovery
  Wie kann die Granularität eines notwendigen Backups oder Recoverys möglichst klein gehalten werden?
- Administration
  Welche Möglichkeiten gibt es, die Administration von umfangreichen Anwendungen transparent zu machen und notwendige Reorganisationen zu vereinfachen?
- Anwendungsgruppierung
  Wie kann man Anwendungen gruppieren, um sie einfacher verwalten oder um den Anwendern Ressourcen in Rechnung stellen zu können?

Sicher gibt es je nach Unternehmen noch weitere Fragestellungen, die in ein Layout der Tablespaces einfließen müssen. Aber man kann schon erkennen, dass die unterschiedlichen Anforderungen sich teilweise ausschließen bzw. aus Sicht der heute üblichen Hardware schwer zu implementieren sind. Geht man z.B. von heute üblichen Festplattenkonfigurationen aus (Festplattenkapazitäten von mehr als 100 GB sind keine Seltenheit), so muss man sich fragen, wie man zum Beispiel die Verfügbarkeitsanforderung bei einer Datenbankgröße von weniger als 10 GB erfüllen kann.

Für viele Unternehmen sind deshalb Raid 5- oder Raid S-Systeme (HP/UX auch Autoraid) das Maß aller Dinge bezüglich ihrer Verfügbarkeit. Bei großen Datenbanken mit mehreren 100 GB Datenvolumen verwendet man logische Devices, bei denen die physikalischen Platten mittels Raid 0+1 zunächst einmal gespiegelt (Raid 1) und dann über mehrere Controller in Stücken à 64 KB oder 128 KB »gestriped« (Raid 0) werden. Dadurch ist ein Optimum an Verfügbarkeit und IO-Verteilung (je nach Anzahl der Controller und Platten) möglich.

Die Frage, die sich dabei stellt, ist: Braucht man in solchen Fällen dann überhaupt noch mehrere Tablespaces? Die Antwort ist: »Ja!«

Zum einen muss das Data Dictionary geschützt werden, um ein unkontrolliertes Wachstum und damit einen eventuellen Absturz der Datenbank zu vermeiden. Daher gehören keine wie auch immer gearteten Benutzerdaten in diesen Tablespace!

Des Weiteren stellt auch ein Raid 5- oder Raid 1-System keine Garantie für hohe Verfügbarkeit dar. Ein Fehler in einem Controller kann z.B. dazu führen, dass Teile des Systems korrumpiert werden. Die Punkte 3 und 4 in der obigen Aufzählung sind auch bei solchen Systemen nicht zu vernachlässigen; sofern der Leser nicht selber DBA ist: Der DBA wird es Ihnen danken!

Folgende Tablespaces sollten deshalb bei jeder Datenbank zusätzlich zu den Tablespaces `SYSTEM` und `SYSAUX` (ab Oracle 10g) angelegt werden. Die Tablespace-Namen sind als Vorschläge gedacht und haben sich in der Praxis gut bewährt:

1. Temporär-Tablespace, z.B. `temp`
2. Undo-Tablespace, z.B. `undotbs`
3. Tablespace für Werkzeuge wie den Oracle Enterprise Manager, z.B. `tools`
4. Mindestens je ein Tablespace für die Daten einer Anwendung, z.B. `xyz_data` für eine Anwendung XYZ
5. Mindestens je ein Tablespace für die Indizes einer Anwendung, z.B. `xyz_index` für eine Anwendung XYZ
6. Eventuell ein Tablespace für Tests und Benutzerdaten, z.B. `users`
7. Eventuell ein oder mehrere Tablespaces für ungewöhnlich große Objekte, z.B. `xyz_data_big`, `xyz_index_big` für eine Anwendung XYZ

Je nach Datenbank können zusätzliche Tablespaces notwendig sein, z.B. beim Einsatz von Partitionen.

3.3.2 Erstellen eines Tablespaces

Mit dem folgenden Befehl wird ein Tablespace für die Anwendung Kundenverwaltung angelegt. Die Werte beziehen sich auf eine Unix-Installation, d.h., die Angabe der Klausel `DATAFILE` muss je nach Betriebssystem angepasst werden.

```
SQL> CREATE SMALLFILE TABLESPACE kunden_daten
     DATAFILE '/oradata1/PDWH10G/kunden_daten01.dbf' SIZE 1000M
     AUTOEXTEND ON NEXT 100M MAXSIZE 5000M
     BLOCKSIZE 8K
     LOGGING
     EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M
     SEGMENT SPACE MANAGEMENT AUTO;
```

Listing 3.1: CREATE TABLESPACE-Kommando

Die Klauseln im Einzelnen:

`TABLESPACE`

Um einen Tablespace anzulegen, sollte man sich zunächst überlegen, welche Aufgabe es zu erfüllen hat, d.h., ob es der Speicherung von Tabellen und Indizes, der Verwaltung von Temporärsegmenten oder dem Undo-Management dient. Es bietet sich an, dem Tablespace einen Namen zu geben, mit dem er eindeutig seiner

Bestimmung zugeordnet werden kann, also z.B. `lager_data`, um zu zeigen, dass es sich hierbei um die Daten der Anwendung LAGERVERWALTUNG handelt. Dazu würde noch ein Tablespace `lager_index` mit den zugehörigen Indizes gehören. Die Länge des Namens ist auf 30 Zeichen begrenzt. Wie bei allen Oracle Data-Dictionary-Einträgen wird standardmäßig nicht zwischen Groß- und Kleinschreibung unterschieden. Soll der Tablespace-Name explizit aus Kleinbuchstaben bestehen, muss der Name mit doppelten Hochkommata angegeben werden. Wie bereits erwähnt, warnen wir allerdings davor, dies zu tun.

`DATAFILE`

Der Name der zugehörigen Datendatei sollte den Namen des Tablespaces beinhalten. Zusätzlich können die Datendateien eines Tablespaces durchnummeriert werden, und sie erhalten eine Endung, die sie auf der Ebene des Betriebssystems als zur Datenbank gehörende Datei kennzeichnet.[2] Dies ist in den meisten Fällen die Endung `.dbf`, allerdings ist auch die Endung `.ora` gebräuchlich. Da Oracle mit den Oracle Managed Files entsprechende Vorschläge macht, werden diese im Folgenden verwendet. Die Endung `.dbf` steht demnach für eine Datendatei, die Endung `.tmp` für eine Temporärdatei.

Oracle Managed Files

Mit dem Serverparameter `db_create_file_dest` ist es ab Oracle9i möglich, ein Verzeichnis für die Erstellung von Datendateien vorzugeben, deren Name beim `CREATE TABLESPACE` oder `ALTER TABLESPACE ADD DATAFILE` nicht angegeben wird. Als Größe wird dabei 100 MB gewählt, wenn sie nicht explizit angegeben ist. Zusammen mit der seit Oracle9i möglichen Nutzung von Locally Managed Tablespaces ergibt sich damit eine erhebliche Vereinfachung bei der Erstellung von Tablespaces bzw. Redolog-Dateien.

Im folgenden Beispiel wird veranschaulicht, wie ein Tablespace ohne Angabe von Lage und Größe der Datendateien angelegt werden kann. Der erste Befehl kann natürlich entfallen, wenn `db_create_file_dest` als Serverparameter gesetzt ist. Der Serverparameter `db_unique_name` sei auf `PDWH10G` gesetzt.

```
SQL> ALTER SYSTEM SET db_create_file_dest = '/oradata1/';
SQL> CREATE TABLESPACE demo_data
     DATAFILE AUTOEXTEND ON NEXT 10M MAXSIZE 500M;
```

Auf die Angabe `DATAFILE AUTOEXTEND` sollte generell nicht verzichtet werden, da ansonsten die Datei beliebig wachsen kann.

Dieser Befehl legt in diesem Beispiel einen Tablespace `demo_data` an mit der Datendatei:

```
/oradata1/PDWH10G/datafile/o1_mf_demo_dat_1g4o4ycv_.dbf
```

Zu beachten ist, dass der Name des Tablespaces in der Datendatei auf acht Zeichen beschränkt wird, um ausreichend Platz für Prä- und Postfix zu behalten.

Für nähere Informationen über Oracle Managed Files siehe Kapitel 3.9.

2 Endungen von Datenbankdateien haben weder Standardwerte noch sind sie Pflicht. Wir empfehlen jedoch wegen der besseren Lesbarkeit, die angegebenen oder ähnliche Endungen zu verwenden.

`SIZE`

Mit einer initialen Größe (`SIZE`) wird Platz auf der Festplatte bzw. dem Filesystem angegeben. Da Oracle den Bereich sofort in Oracle-Blöcke einteilt, kann das Erstellen eines Tablespaces durchaus mehrere Minuten betragen.

Bei der Verwendung von Oracle Managed Files wird eine Standardgröße von 100 MB angenommen, die jedoch jederzeit geändert werden kann. Es gibt je nach Betriebssystem maximale Größen von Dateien, allerdings liegt diese bei den neueren 64-Bit- und auch einigen 32-Bit-Betriebssystemen bei mehr als 2 GB bis hin zu 256 GB.

`SMALLFILE`

Diese Klausel ist neu in Oracle 10g und erleichtert die Verwaltung sehr großer Datenbanken. Bis einschließlich Oracle9i ist nämlich die Maximalgröße einer Datendatei (unabhängig von Betriebssystemgrenzen) von der verwendeten Oracle-Blockgröße des Tablespaces abhängig, die Obergrenze liegt konkret bei 4M Blöcken. Für eine Blockgröße von 8 KB ergibt sich somit eine maximale Datendateigröße von 32 GB. Über diese Grenze hinaus kann der Tablespace nur durch Hinzufügen weiterer Datendateien vergrößert werden.

Unter Oracle 10g gibt es nun einen Ausweg: die so genannten `BIGFILE`-Tablespaces können bis zu 4G Blöcke umfassen, kommen bei einer Blockgröße von 8 KB also auf bis zu 32 TB Datendateigröße.

Dies ist aber nur für Umgebungen mit Volume-Managern zu empfehlen, die ein Striping oder RAID leisten, auch vor dem Hintergrund, dass ein solcher Tablespace auf eine einzige Datendatei beschränkt bleibt. Die Möglichkeit, die logischen Volumes dynamisch – also im laufenden Betrieb – zu erweitern, sollte ebenfalls Voraussetzung für die Verwendung solcher Tablespaces sein.

`BIGFILE`-Tablespaces sind nur als Locally Managed Tablespaces mit automatischer Segmentplatzverwaltung (ASSM) möglich (s.u.), hiervon ausgenommen sind das System-Tablespace, Temporär-Tablespaces und Undo-Tablespaces. Bei diesen Tablespaces ist die Bigfile-Option auch mit manueller Segmentverwaltung möglich.

Um kompatibel mit existierenden Skripten zu bleiben, ist die Klausel `SMALLFILE` die Standardeinstellung nach Erstellung einer Datenbank und hätte im obigen Beispiel somit weggelassen werden können. Durch die Klausel `BIGFILE` wird entsprechend ein `BIGFILE`-Tablespace erzeugt. Es ist sogar möglich, die Standardeinstellung der Datenbank zu ändern:

```
SQL> ALTER DATABASE SET DEFAULT BIGFILE TABLESPACE;
SQL> CREATE TABLESPACE bigts …;
```

Da ein `BIGFILE`-Tablespace nur aus einer einzigen Datendatei bestehen kann, ist für Größenänderungen dieser Datendatei nur der Tablespace-Name erforderlich, z.B.:

```
SQL> ALTER TABLESPACE bigts RESIZE 200M;
```

Es muss auch berücksichtigt werden, dass im Falle einer Beschädigung und anschließenden Wiederherstellung eines Tablespaces die unterste Granularitätsstufe, nämlich Recovery einer einzelnen Datendatei, mit `BIGFILE`-Tablespaces grundsätzlich identisch zum Recovery des gesamten Tablespaces ist. Bei großen Tablespaces kann dies erheblich länger dauern als das Recovery einer oder weniger Datendateien des betroffenen Tablespaces.

AUTOEXTEND

Mit der Option AUTOEXTEND ON kann eine automatische Vergrößerung der Datendatei nach Bedarf erfolgen. In diesem Falle wird, wenn der Platz in der Datendatei nicht mehr ausreicht, diese Datei um die Größe NEXT vergrößert – bis zu einer maximalen Größe von MAXSIZE. Wird der Parameter NEXT nicht angegeben, wird die Datei jeweils um einen einzigen Datenbankblock vergrößert! (Bei Oracle Managed Files immer um 100 MB.) Dies kann zu erheblichen Performance-Einbußen führen, so dass anzuraten ist, den Parameter NEXT nicht zu klein zu wählen (10 MB oder größer) und die Initialgröße von vornherein ausreichend zu dimensionieren.

Sollten Sie mehrere Datendateien in einem Dateisystem anlegen, müssen Sie darauf achten, dass jede mit AUTOEXTEND angelegte Datei unter Umständen Platz reserviert, den Sie eigentlich für eine andere Datei vorgesehen hatten. Daher empfiehlt es sich, die maximale Größe der Datei mit dem Parameter MAXSIZE zu beschränken. Standardmäßig wird dieser Wert auf UNLIMITED gesetzt, so dass jede Datei beliebig viel Platz reservieren kann.

Auch an diesen Parametern sieht man, wie wichtig die Aufteilung von Anwendungen auf unterschiedliche Tablespaces ist, da es nur so möglich ist, bestimmte Anwendungen auf ihren Plattenplatz hin zu limitieren.

BLOCKSIZE

Mit dieser Klausel ist es möglich, einen Tablespace mit einer anderen Blockgröße als der Standardblockgröße der Datenbank anzulegen. Damit kann zum Beispiel für die Abspeicherung von Binärdaten in BLOB-Feldern ein spezieller Tablespace erstellt werden, der den Anforderungen dieser Felder entgegenkommt und somit eine Verkettung von Blöcken vermeidet oder zumindest minimiert. Außerdem können hierdurch die transportablen Tablespaces besser unterstützt werden, da jetzt in jeder Zieldatenbank ein Tablespace mit identischer Blockgröße aufgebaut werden kann.

Um eine derartige Blockgröße nutzen zu können, müssen der zugehörige Serverparameter db_*n*k_cache_size und der Parameter db_cache_size gesetzt sein, damit im Buffer-Cache entsprechende Ressourcen zur Verfügung stehen. Es versteht sich daher von selbst, dass nur die Größen als BLOCKSIZE erlaubt sind, für die es einen entsprechenden Serverparameter gibt. Dies sind: 2 KB, 4 KB, 8 KB, 16 KB und 32 KB.

LOGGING

Die Klausel LOGGING steht an dieser Stelle für das Standardverhalten der Objekte, die in dem Tablespace gespeichert werden. Logging besagt, dass wie üblich alle DML-Operationen über den Redolog-Mechanismus protokolliert werden. Alternativ kann durch ein NOLOGGING erreicht werden, dass bestimmte DML-Operationen (so genannte *Direct Loads*, beispielsweise durch den Oracle SQL*Loader) nicht mitprotokolliert werden. Das bedeutet, falls ein Media-Fehler auftritt, können die geladenen Daten nicht wiederhergestellt werden, und die zugehörige Data-Dictionary-Information (z.B. Extent-Verwaltung) wird als logisch korrupt gekennzeichnet.

`EXTENT MANAGEMENT LOCAL`

Diese Klausel definiert den Tablespace als so genannten Locally Managed Tablespace (LMTS). Bei dieser Methode verwaltet der Tablespace seine Extents über eine lokale Bitmap-Struktur in jeder zugehörigen Datendatei. Seit Oracle8i ersetzt diese Methode zunehmend die alte Variante der so genannten Dictionary Managed Tablespaces (DMTS). Die Extent-Allokierung wird beschleunigt, und die früher üblichen rekursiven Operationen entfallen, es muss auch keine Rollback-Information erzeugt werden. Darüber hinaus wird die Verwaltung der Extents vereinfacht, so dass ein Zusammenlegen (Coalesce) von freien Extents entfallen kann.

LM-Tablespaces sind mittlerweile die Standardeinstellung und haben sich in der Praxis ausnahmslos bewährt. Die alten DM-Tablespaces sollten nicht mehr verwendet werden. Damit entfällt auch die Notwendigkeit, die früher üblichen Klauseln für Default-Storage-Parameter sowie für eine minimale Extent-Größe anzugeben.

Bei der Angabe der Extent-Größe für Objekte gibt es jetzt nur noch zwei Alternativen: `UNIFORM SIZE` legt eine einheitliche Größe für alle Extents fest, `AUTOALLOCATE` (Standard) kalkuliert die Größe der Extents. Dabei werden die Extents in der Regel wie folgt angelegt.

- 64 KB für die ersten 16 Extents
- 1 MB ab dem 17. Extent
- 8 MB ab dem 80. Extent
- 64 MB ab dem 200. Extent

Die tatsächlich allokierten Extent-Größen können von diesem Schema abweichen, insbesondere z.B. bei der Nutzung von `ALTER TABLE <tab> ALLOCATE EXTENT`.

Die Benutzung von `AUTOALLOCATE` empfiehlt sich fast immer; auch große Objekte können hiermit effizient behandelt werden. Lediglich für extrem große Objekte sollten Tablespaces mit `UNIFORM SIZE` benutzt werden.

```
SQL> CREATE TABLESPACE kunden_daten_small
     DATAFILE '/oradata1/PDWH1OG/kunden_daten_small01.dbf'
     SIZE 100M AUTOEXTEND ON NEXT 10M MAXSIZE 300M
     EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

Listing 3.2: Locally Managed Tablespace mit `AUTOALLOCATE`

`SEGMENT SPACE MANAGEMENT`

Die Verwaltung der freien Blöcke für ein Segment geschah bis einschließlich Oracle8i durch sog. Freilisten, d.h. verkettete Listen, die alle freien Blöcke diesseits der *High Water Mark* enthalten. Diese Listen werden schubweise beim Allokieren eines Extents gefüllt, und die `INSERT`-Operationen holen sich dann den nächsten freien Block aus dieser Liste. Wird eine bestimmte Marke (`PCTFREE`) in einem Block überschritten, dann wird dieser von der Freiliste gelöscht. Wird durch Änderungs- oder Löschoperationen eine zweite Marke (`PCTUSED`) in einem Block unterschritten, dann wird der entsprechende Block wieder in die Freilisten eingetragen.

Bei großer Transaktionslast, speziell mit INSERT- und DELETE-Operationen, kann eine solche Freiliste zu einem Engpass führen. Dadurch kommt es zu Wartezuständen bei der Verarbeitung, die in der Vergangenheit nur durch mehrere Freilisten ausgeglichen werden konnten. Als Tuning-Maßnahme war es üblich, die Parameter FREELISTS und FREELIST GROUPS hoch zu setzen, damit nicht alle INSERT-Operationen über die gleiche Freiliste abgewickelt werden. Die Änderung des Parameters FREELIST GROUPS kann jedoch nur durch eine Reorganisation der gesamten Tabelle erfolgen, wohingegen FREELISTS auch durch einen ALTER TABLE-Befehl angepasst werden kann.

Seit Oracle9i und bei Verwendung von Locally Managed Tablespaces kann durch die Klausel SEGMENT SPACE MANAGEMENT AUTO auf eine Verwaltung über Freilisten verzichtet werden. Bei dieser Methode (ASSM für *Automatic Segment Space Management* genannt) wird über eine Bitstruktur der freie Platz in den Blöcken verwaltet. Die Bitstruktur ist je nach Segmenttyp (Daten, Index, LOB) unterschiedlich, hat aber generell folgende Struktur:

- Das erste Bit zeigt an, ob der Block als voll markiert wurde; dies geschieht, wenn der Wert PCTFREE überschritten wurde.
  - 1 = voll
- Das zweite und dritte Bit zeigen den Füllgrad des Blockes an (nur Datensegmente).
  - 00 = freier Platz größer 75 Prozent
  - 01 = freier Platz größer 50 Prozent
  - 10 = freier Platz größer 25 Prozent
  - 11 = freier Platz kleiner 25 Prozent
- Das letzte Bit zeigt an, ob der Block formatiert wurde.
  - 1 = formatiert

Das zweite und dritte Bit ersetzen zusätzlich den Parameter PCTUSED, d.h., ein Block wird wieder mit Sätzen gefüllt, wenn der Platz unter die PCTFREE-Marke gesunken ist.

Bei Indizes entfällt der Füllgrad der Blöcke, da es hier nur entscheidend ist, ob ein Block formatiert oder voll ist.

Die Verwaltung des freien Platzes für ein Segment erfolgt jetzt in bis zu drei Stufen. Die erste Stufe bilden Bitmapped-Blöcke, die an den Segment-Header gebunden sind, eine Zwischenstufe von Bitmapped-Blöcken wird für große Tabellen verwendet. Die eigentliche Verwaltung in den Blöcken erfolgt dann in der dritten Stufe. Hier werden je Bitmapped-Block mehrere freie Blöcke verwaltet.

Bei INSERT-Operationen wird jetzt vom Segment-Header aus geprüft, ob in einem Block noch genügend Platz vorhanden ist. Ist dies der Fall, wird der Datensatz eingefügt, und wenn ein Schwellenwert (25%, 50%, 75%) überschritten wurde, wird die entsprechende Bitstruktur angepasst. Wenn kein freier Block mehr zur Verfügung steht, werden neue Blöcke formatiert (je nach Verfahren mindestens 16 Blöcke), bzw. ein neues Extent wird allokiert. Wenn viele INSERT-Operationen (z.B. Parallel INSERT) ausgeführt werden, wird die Formatierung der Blöcke auf mehrere

Prozesse über einen Hash-Algorithmus verteilt. Dadurch kann es vorkommen, dass bestimmte Blöcke, obwohl sie unterhalb der *High –Water Mark* liegen, nicht formatiert werden. Dies ist ein kleiner Nachteil dieser Freiplatzverwaltung, da u.U. Platz verschwendet wird (natürlich werden die Blöcke bei einer nächsten Formatierung mit benutzt).

Der Vorteil des Verfahrens ist aber die geringe Latch-Contention (Sperren auf der Freelist eines Segmentes) und schnellere Vergabe von freiem Platz.

Nach unseren Erfahrungen kann die automatische Segmentplatzverwaltung in den meisten Fällen uneingeschränkt empfohlen werden. Eine Ausnahme bilden jedoch Tablespaces, in denen LOB-Segmente gespeichert werden, da die Kombination von LOB-Segmenten und ASSM sowohl in Oracle9i als auch in Oracle 10g Gegenstand zahlreicher Bugs im Zusammenhang mit der Speicherplatzverwaltung war und ist. Hier muss sich letztlich zeigen, ob diese Problematik mit den nächsten Software-Patches entschärft wird.

Bestimmte Anwendungen können aber nach wie vor von der manuellen Segmentplatzverwaltung profitieren. Hierzu gehören solche Anwendungen, in denen von Zeit zu Zeit größere Datenmengen gelöscht werden, ohne dass komplette Blöcke geleert werden. Bei der klassischen Segmentverwaltung führte dies dazu, dass die freien Bereiche unter Umständen nicht wieder verwendet wurden. Mit ASSM werden die neuen Daten jetzt auf alle teilgefüllten Blöcke verteilt. Obwohl dies im ersten Schritt besser erscheinen mag, birgt es einen – u.U. – entscheidenden Nachteil. Neue Daten haben oft eine Beziehung zueinander, z.B. die aktuellen Auftragseingänge. Durch die Verteilung über ASSM müssen in der Regel mehr Blöcke in den Buffer Cache gelesen werden, als es mit der älteren Methode, bei der neue Daten in einem neuen, d.h., leeren Block landen, der Fall gewesen wäre. Wenn Sie Objekte haben, bei denen das der Fall ist, sollten Sie überlegen, ob Sie für diese Objekte einen eigenen Tablespace mit manueller Segmentplatzverwaltung anlegen.

3.3.3 SYSTEM- und SYSAUX-Tablespace

Jedem, der schon einmal eine Oracle-Datenbank administriert hat, wird der `SYSTEM`-Tablespace ein Begriff sein. Seit jeher handelt es sich hierbei um den ersten Tablespace einer Datenbank, in dem sämtliche Metadaten und Verwaltungsinformationen der Datenbank, das so genannte Data Dictionary, abgespeichert werden. Aus diesem Grund ist der `SYSTEM`-Tablespace höchst sensibel und sollte für alle »normalen« Benutzer oder Anwendungsbenutzer tabu sein.

Der SYSAUX-Tablespace hingegen ist neu in Oracle 10g. Genau wie der SYSTEM-Tablespace ist er zwingender Bestandteil einer Oracle 10g-Datenbank. Dies ist insbesondere bei Migrationen von älteren Versionen nach Oracle 10g zu berücksichtigen.

Der SYSTEM-Tablespace

Der `SYSTEM`-Tablespace wird bei der Erstellung der Datenbank generiert. Seine Datendateien müssen über die `DATAFILE`-Klausel des `CREATE DATABASE`-Kommandos angegeben werden, es sei denn, es werden Oracle Managed Files verwendet, so dass die Datendatei mit Standardparametern angelegt werden kann. Wie bei jeder anderen Erstellung eines Tablespaces können bei Bedarf auch mehrere Dateien durch

Kommata getrennt angegeben und die automatische Erweiterbarkeit (Klausel AUTOEXTEND) eingeschaltet werden. Sie sollten unbedingt auch die Klausel EXTENT MANAGEMENT LOCAL verwenden, da im Gegensatz zu anderen Tablespaces der SYSTEM-Tablespace standardmäßig noch als Tablespace des alten Typs (Dictionary-Managed) angelegt wird. Zu beachten ist allerdings, dass Sie dann generell keine Dictionary-Managed Tablespaces mehr anlegen können.

Mit dem Befehl CREATE DATABASE wird das interne Data Dictionary aufgebaut, über das sich die Datenbank verwaltet. Die dem Benutzer SYS gehörenden Tabellen liegen im SYSTEM-Tablespace und werden durch die Endung »$« gekennzeichnet. Es ist nicht erlaubt, DML-Befehle[3] (INSERT, UPDATE, DELETE) auf Data-Dictionary-Tabellen auszuführen; die einzige Ausnahme bildet die Audit-Trail-Tabelle aud$, da sie vom Administrator in regelmäßigen Abständen bereinigt werden muss.

Das interne Data Dictionary wird den Administratoren und Benutzern durch eine Reihe von Sichten (externes Data Dictionary) zur Verfügung gestellt.

Das interne Data Dictionary

Wie bereits erwähnt, verwaltet sich die Datenbank über das Data Dictionary. Genau wie jede Anwendung aus einer Reihe von Objekten besteht, die über Schlüsselwerte in Bezug zueinander stehen, ist auch das Data Dictionary eine Anwendung, die über Schlüssel verwaltet wird. Im Gegensatz zu »normalen« Tabellen wird hierauf jedoch nicht mit DML-Befehlen, sondern mit DDL-Befehlen zugegriffen. Warum dies notwendig ist, zeigt folgendes Beispiel:

```
SQL> CREATE TABLE personen (
       persnr      NUMBER        NOT NULL,
       anrede      VARCHAR2 (5),
       vorname     VARCHAR2 (20),
       nachname    VARCHAR2 (20),
       geburtstag  DATE
     )
     TABLESPACE users;
```

Listing 3.3: Erstellen einer Tabelle

Mit diesem Befehl wird eine Tabelle mit dem Namen personen und fünf Spalten im Tablespace users angelegt. Was als DDL-Befehl CREATE TABLE nur wenige Zeilen benötigt, wird durch den Kernel in 53 (!) Befehle aufgeteilt. Was geschieht mit diesem Befehl?

Zunächst einmal muss überprüft werden, ob der Anwender, der den Befehl ausführt, die entsprechenden Rechte hat, d.h.: »Darf er ein CREATE TABLE-Kommando absetzen?« »Hat er das Recht, im Tablespace users eine Tabelle anzulegen?« Dann muss überprüft werden, ob in dem Tablespace ausreichend Platz für ein Extent mit der vordefinierten Größe vorhanden ist. Außerdem müssen die Speichereigenschaften aus den Standardvorgaben des Tablespaces gewonnen werden. Wenn diese Abfragen erfolgreich waren, können die entsprechenden Einträge im Data Dictionary vorgenommen werden. Dies sind unter anderem Einträge in tab$ für

3 DML steht für *Data Manipulation Language* (INSERT, UPDATE, DELETE), DDL für *Data Definition Language* (CREATE, ALTER, DROP).

den Namen der Tabelle und `col$` für die Spalten der Tabelle. Die folgende Liste zeigt alle Tabellen des Data Dictionarys, in die in diesem Fall Einträge vorgenommen wurden:

```
obj$, seg$, tsq$, con$, tab$, col$, ccol$, cdef$
```

Auffallend ist die Endung »$«, die anzeigt, dass es sich hierbei um die Tabellen des Data Dictionarys handelt, die dem Benutzer `SYS` gehören.

Ähnlich wie die Erstellung einer neuen Tabelle über DDL-Befehle versteckt wird, wird auch die Abfrage von Informationen über das externe Data Dictionary erleichtert. Das Beispiel, um die Struktur einer Tabelle anzuzeigen, verdeutlicht dies:

```
SQL> SELECT * FROM user_tab_columns;
```

Dieser Befehl greift auf folgende Data-Dictionary-Tabellen zu:

```
obj$, user$, col$, coltype$, hist_head$, tab$
```

Es ist also leicht nachzuvollziehen, warum es nicht erlaubt ist, mit DML-Befehlen auf das Data Dictionary zuzugreifen. Die Gefahr einer Inkonsistenz und damit der Korruption der Datenbank ist zu groß!

Ebenso sollten die Views, die auf das Data Dictionary zugreifen, nicht manipuliert werden, allerdings ist hier die Gefahr nicht ganz so groß, da diese über das im Folgenden besprochene SQL-Skript `catalog.sql` jederzeit wiederhergestellt werden können.

Das externe Data Dictionary

Mit dem Skript `catalog.sql` (Verzeichnis: `$ORACLE_HOME/rdbms/admin`) wird die Benutzersicht auf das interne Data Dictionary aufgebaut.

```
sqlplus / as sysdba @?/rdbms/admin/catalog.sql
```

Das »?« steht hierbei für das `ORACLE_HOME`-Verzeichnis und kann sowohl unter Unix als auch MS-Windows verwendet werden.

Dieses Skript baut verschiedene Views auf.

1. Virtuelle Views (`v$, gv$`)

   Dies sind Informationen der Kontrolldateien sowie Zähler und Strukturen der Prozesse und des Shared Memorys.

   Zum Beispiel gibt `v$datafile` die Liste der Datenbankdateien aus Sicht der Kontrolldatei an.

```
SQL> SELECT name,value
     FROM   v$parameter
     WHERE  name LIKE '%undo%';
NAME                           VALUE
------------------------------ ----------------
undo_management                AUTO
undo_tablespace                UNDOTBS1
undo_retention                 7200
```

Listing 3.4: Auszug aus der View `v$parameter`

Globale virtuelle Views sind eine Erweiterung für den RAC(Real Application Clusters)-Betrieb. Sie zeigen die entsprechenden Informationen zusammenfassend für alle Instanzen in einem RAC. Diese Views beginnen mit `gv$` anstelle von `v$`.

2. DBA-Views (`dba_`)

   Dies ist die komplette Repräsentation des internen Data Dictionarys, d.h., über diese Views können alle relevanten Informationen über die Datenbank abgefragt werden. Die Informationen können nur von Benutzern gesehen werden, die zur Gruppe der Administratoren gehören bzw. die Rolle `select_catalog_role` zugeteilt bekommen haben.

   Als Beispiel die Abfrage der Datenbankdateien (`dba_data_files`) aus Sicht des Data Dictionarys:

```
SQL> SELECT file_name, tablespace_name, bytes
     FROM   dba_data_files;

FILE_NAME                              TABLESPA      BYTES
-------------------------------------- -------- ----------
D:\ORACLE\ORADATA\PS10\SYSTEM01.DBF     SYSTEM    272629760
D:\ORACLE\ORADATA\PS10\UNDOTBS01.DBF    UNDOTBS1  209715200
D:\ORACLE\ORADATA\PS10\SYSAUX01.DBF     SYSAUX    209715200
D:\ORACLE\ORADATA\PS10\INDX01.DBF       INDX      209715200
D:\ORACLE\ORADATA\PS10\USERS01.DBF      USERS     209715200
D:\ORACLE\ORADATA\PS10\TOOLS01.DBF      TOOLS     104857600
```

Listing 3.5: Die View dba_data_files

3. Views für alle Benutzer (`all_`)

   Diese Views erlauben es den Benutzern, auf Objekte zuzugreifen, auf denen ihnen Rechte gegeben wurden. Dazu gehören natürlich auch die eigenen Objekte.

   Als Beispiel wird hier mit der View `all_tables` eine Liste aller lesbaren Tabellen für den momentan angemeldeten Benutzer (in diesem Fall `demo`) erstellt.

```
SQL> SELECT owner, table_name
     FROM   all_tables;

OWNER      TABLE_NAME
---------- --------------------------
SYS        SCHEDULER$_JOB_STEP_STATE
SYS        AUDIT_ACTIONS
SYS        STMT_AUDIT_OPTION_MAP
SYS        TABLE_PRIVILEGE_MAP
SYS        SYSTEM_PRIVILEGE_MAP
SYS        DUAL
SYS        PLAN_TABLE$
SYS        KU$NOEXP_TAB
SYS        ODCI_WARNINGS$
SYS        ODCI_SECOBJ$
```

```
SYS        PSTUBTBL
SYSTEM     HELP
SYSTEM     DEF$_TEMP$LOB
SYSTEM     OL$NODES
SYSTEM     OL$HINTS
SYSTEM     OL$
WMSYS      WM$VERSION_HIERARCHY_TABLE
WMSYS      WM$NEXTVER_TABLE
WMSYS      WM$VERSION_TABLE
WMSYS      WM$WORKSPACES_TABLE
DEMO       AUFPOSITIONEN
DEMO       PRODUKTE
DEMO       AUFTRAEGE
DEMO       AUFSTATUS
DEMO       TELNUMMERN
DEMO       ADRESSEN
DEMO       TYPEN
DEMO       KUNDEN
```

Listing 3.6: Die View `all_tables` ***für den Benutzer*** `demo`

Für den Datenbankadministrator besteht oftmals kein Unterschied zwischen den `dba`- und den `all`-Views.

4. Views nur für den lokalen Benutzer (`user_`)

   Hier werden nur noch die Informationen zu Objekten des momentan angemeldeten Benutzers angezeigt. Als Beispiel hier die Sicht `user_tables`, die im Gegensatz zu `all_tables` keine Information über den Besitzer (`owner`) anzeigt.

```
SQL> SELECT table_name
     FROM   user_tables;

TABLE_NAME
--------------------------
KUNDEN
TYPEN
ADRESSEN
TELNUMMERN
AUFSTATUS
AUFTRAEGE
PRODUKTE
AUFPOSITIONEN
```

Listing 3.7: Die View `user_tables` ***für den Benutzer*** `demo`

Zusätzlich zu dem Skript `catalog.sql` gibt es weitere Skripte, die das Data Dictionary für unterschiedliche Anwendungen (XML, Java etc.) erweitert (siehe Kapitel 2).

Der SYSAUX-Tablespace

Im Laufe der Oracle-Versionen ist eine ganze Reihe von erweiterten Funktionen entstanden, die zwar zum Teil eigenständige Schemata belegen, ihre Objekte aber dennoch im SYSTEM-Tablespace speichern, z.B. die Verwaltung von Stored Outlines (Schema `OUTLN`), der Workspace-Manager (Schema `WMSYS`), der Log Miner usw. Dies ist ungünstig, da diese Funktionen eine gewisse Dynamik mit sich bringen, die im sensiblen SYSTEM-Tablespace fehl am Platze ist.

Andere Funktionen wie z.B. die XML DB oder das Performance-Werkzeug Statspack konnten ihre Daten in einem benutzerdefinierten Tablespace speichern. Oracle 10g kommt schließlich mit einigen neuen Funktionen, die insbesondere eine Archivierung diverser Performance-Statistiken beinhalten (Stichwort AWR, siehe auch Kapitel 11.2), was zu erheblichem Platzbedarf führen kann. Man hat daher die Gelegenheit genutzt, mit dem Wildwuchs diverser Daten-Repositories aufzuräumen und stattdessen einen einheitlichen Standard-Tablespace namens `SYSAUX` hierfür zu benutzen.

Der SYSAUX-Tablespace wird beim Erstellen der Datenbank angelegt, seine Datendatei(en) müssen über die `SYSAUX DATAFILE`-Klausel angegeben werden (wie beim SYSTEM-Tablespace können natürlich auch hier Oracle Managed Files verwendet werden).

Die View `v$sysaux_occupants` gibt Auskunft darüber, welche Komponenten der Datenbank den SYSUAX-Tablespace benutzen und wie viel Platz jede einzelne Komponente belegt. Für einige Komponenten besteht auch die Möglichkeit, sie in einen alternativen Tablespace zu migrieren. Das Verfahren dafür ist über die Spalten `move_procedure` und `move_procedure_desc` beschrieben. Als Startwert sollte für den SYSAUX-Tablespace ein Platzbedarf zwischen 500 MB und wenigen GB gewählt werden.

Im Rahmen eines Upgrades von älteren Datenbankversionen muss der SYSAUX-Tablespace erstellt werden. Dies muss nach dem Hochfahren der Datenbank mittels `STARTUP UPGRADE`, aber noch vor dem Aufruf des Upgrade-Skripts (z.B. `u0902000.sql`) geschehen. Für Näheres siehe *Oracle 10g Database Upgrade Guide*.

3.3.4 Temporär-Tablespaces und Temporärsegmente

In einem DBMS fallen immer wieder Sortier- oder Hash-Operationen oder damit verwandte Operationen wie Gruppierungen oder Index-Aufbau an. Diese Operationen benötigen Speicherplatz für Zwischenergebnisse und das Endergebnis. Man bezeichnet diesen Speicher als Temporärspeicher.

Falls nur eine kleine Menge an Temporärspeicherplatz benötigt wird, wird im Oracle Server ein so genannter Sortierbereich benutzt. Dieser befindet sich im Hauptspeicher des Datenbanksystems und wird über den Serverparameter `sort_area_size` explizit oder `pga_aggregate_target` als Gesamtheit festgelegt. Werden allerdings größere Mengen Speicherplatz angefordert, so wird automatisch auf Temporärsegmente ausgewichen, d.h., es wird ein spezielles Segment in der Datenbank reserviert, das Zwischenergebnisse und das Endergebnis aufnimmt.

Im Unterschied zu allen anderen Segmenttypen werden Temporärsegmente grundsätzlich implizit angelegt und können nicht explizit angesprochen oder ausgelesen werden. Die bisher einzige Ausnahme hierfür sind globale Temporärtabellen, deren Inhalte ebenfalls als Temporärsegmente abgelegt werden. Nach erfolgreicher Durchführung der Sortierung oder Gruppierung werden Temporärsegmente wieder freigegeben; der Inhalt von globalen Temporärtabellen wird je nach Definition nach Transaktions- bzw. Sitzungsende freigegeben.

Durch den mit Oracle8i eingeführten Befehl `CREATE TEMPORARY TABLESPACE` ist es möglich, einen so genannten Temporär-Tablespace speziell für die Aufnahme von Temporärsegmenten aufzubauen. Entscheidend ist, dass die gesamte Verwaltung eines solchen Tablespaces ohne Beteiligung des Data Dictionarys stattfindet. Da Dateien eines Temporär-Tablespaces (so genannte Temporärdateien) in einem separaten Bereich in der Kontrolldatei gespeichert werden, »sieht« ein Checkpoint oder ein Backup über den Recovery-Manager diese Tablespaces nicht. Somit ist auch ein Restore nach einem Fehler nicht notwendig – man erstellt die Temporärdateien einfach neu.

```
SQL> CREATE TEMPORARY TABLESPACE temp
     TEMPFILE '/oradata1/PDWH10G/temp01.tmp' size 1000M
     AUTOEXTEND ON NEXT 100M MAXSIZE 5000M
     EXTENT MANAGEMENT LOCAL UNIFORM SIZE 10M;
```

Listing 3.8: Anlegen eines Temporär-Tablespaces

Mit diesem Befehl wird ein Tablespace mit Namen `temp` erstellt und durch die Angabe `EXTENT MANAGEMENT LOCAL` als Locally Managed Tablespace verwaltet. Auch hier ist die Angabe `EXTENT MANAGEMENT LOCAL` optional, da es zurzeit keine andere Möglichkeit gibt. Um Temporärdateien auch betriebssystemseitig von Datendateien unterscheiden zu können, empfiehlt es sich, sie mit der Endung `.tmp` anzulegen.

Die Erstellung von Temporärdateien läuft unabhängig von ihrer Größe im Sekundenbereich ab, da für die Temporärdateien im Gegensatz zu Datendateien lediglich Plattenplatz reserviert, aber keine Oracle-Blockformatierung durchgeführt wird.

Jedem Datenbankbenutzer ist ein Temporär-Tablespace zugewiesen, in dem vom Oracle-Server für umfangreiche Sortieroperationen Temporärsegmente angelegt werden. Findet keine explizite Zuweisung statt, wird dem Benutzer standardmäßig der `SYSTEM`-Tablespace als Temporär-Tablespace zugeordnet. Dies sollte natürlich nicht passieren. Um zu verhindern, dass dies versehentlich passiert, kann ein Temporär-Tablespace als Default –Temporary Tablespace für die Datenbank ausgewählt werden:

```
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp;
```

Diese Zuweisung kann auch gleich bei der Erstellung der Datenbank im Rahmen des `CREATE DATABASE`-Kommandos geschehen.

Wichtig zu wissen ist, dass der Default-Temporär-Tablespace einer Datenbank – wenn er erst einmal als solcher definiert ist – nicht mehr gelöscht werden kann, bevor nicht ein anderer Temporär-Tablespace seine Rolle übernommen hat. Damit wird verhindert, dass die Benutzer auf den `SYSTEM`-Tablespace »zurückfallen«.

Temporär-Tablespace-Gruppen

Mit Oracle 10g ist es möglich, mehrere Temporär-Tablespaces zu einer Gruppe zusammenfassen und eine solche Gruppe wie oben beschrieben einem Benutzer zuzuordnen oder datenbankweit als Standard zu definieren. Dies ist vor allem im Hinblick auf bessere I/O-Verteilung beim Umgang mit Temporärsegmenten gedacht, da nun zufällig ausgewählt wird, in welchem Tablespace ein Temporärsegment erstellt wird. Die Temporärdateien der einzelnen Tablespaces sollten also möglichst verteilt liegen, um eine echte I/O-Parallelisierung zu erreichen. In Umgebungen mit Striping oder RAID 5 ist eine Parallelisierung ohnehin gegeben, so dass der Vorteil hier eher gering ist.

Eine Tablespace-Gruppe wird nicht explizit definiert, sondern automatisch angelegt, wenn ein Tablespace der Gruppe zugeordnet wird, z.B.:

```
SQL> ALTER TABLESPACE temp TABLESPACE GROUP temp_group;
SQL> ALTER TABLESPACE temp2 TABLESPACE GROUP temp_group;
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp_group;
SQL> ALTER USER demo TEMPORARY TABLESPACE temp_group;
```

Auch hier gilt, dass – sofern eine Tablespace-Gruppe als datenbankweiter Default definiert ist – kein Mitglied dieser Gruppe gelöscht werden kann.

Um einen Tablespace aus einer Gruppe zu entfernen, wird er der leeren Gruppe `''` zugewiesen:

```
SQL> ALTER TABLESPACE temp TABLESPACE GROUP '';
```

Auch das Löschen einer Tablespace-Gruppe geschieht nicht explizit, sondern automatisch, sobald der letzte Tablespace aus ihr entfernt wird. Vorhandene Tablespace-Gruppen können über die View `dba_tablespace_groups` eingesehen werden.

Achtung: Tablespaces und Tablespace-Gruppen teilen sich einen gemeinsamen Namensraum, können also nicht identische Namen haben.

3.3.5 Undo-Tablespaces

Ein Undo-Tablespace beinhaltet ausschließlich Undo-Segmente. Primäre Aufgabe der Undo-Segmente ist es, den alten Zustand von Feldinhalten (*before images*) so lange zu speichern, bis der neue Zustand in der Datenbank festgeschrieben wurde. Dadurch kann sichergestellt werden, dass Befehle entweder vom System (PMON-Prozesss) oder vom Anwender wieder zurückgenommen werden können. Solche Situationen können entstehen, wenn interne Ressourcen aufgebraucht sind, z.B. nicht genug Platz in einem Tablespace vorhanden ist, oder wenn ein Anwender mit dem expliziten Befehl `ROLLBACK` eine Transaktion rückgängig macht.

Für Zugriffe auf das Data Dictionary gibt es ein spezielles Undo-Segment `SYSTEM` im gleichnamigen Tablespace `SYSTEM`. Da dieses ausschließlich für Transaktionen im Tablespace `SYSTEM` – und damit in der Regel für DDL-Befehle – reserviert ist, benötigt man für eine produktive Datenbank weitere Undo-Segmente, die zyklisch beschrieben werden.

Seit Oracle9i gibt es die Möglichkeit, die Instanz mit automatischer Undo-Verwaltung zu betreiben (Serverparameter `undo_management = AUTO`). Sowohl für Oracle9i als auch für Oracle 10g ist diese Technik der herkömmlichen, manuellen Undo-Verwaltung vorzuziehen, daher wird hier auch nur dieser Fall betrachtet. Für ältere Datenbanken und Migrationsmöglichkeiten wird auf Kapitel 3.10 verwiesen.

Bei automatischer Undo-Verwaltung werden die notwendigen Segmente vom System in einem speziellen Undo-Tablespace automatisch angelegt, der Administrator hat keinerlei Einfluss auf Anzahl und Größe der Undo-Segmente.

Die interne Verwaltung der Undo-Segmente erfolgt in Form von Ringpuffern. Jedes Undo-Segment kann eine oder mehrere Transaktionen beherbergen, auch kann eine einzelne Transaktion sich durchaus über mehrere Undo-Segmente erstrecken. Jedes Extent eines Undo-Segmentes kann ebenfalls eine oder mehrere Transaktionen speichern. Es versteht sich, dass eine Transaktion umso mehr Speicherplatz belegt, je umfangreicher ihre Änderungen ausfallen. Undo-Segmente sind – wie bereits erwähnt – als logische »Ringe« aufgebaut sind. Ist ein Extent gefüllt, wird entweder

- das nächste, bereits existierende Extent wieder verwendet, aber nur dann, wenn es keine aktiven Transaktionen mehr enthält, oder
- eine neues Extent angelegt und logisch in den Ring eingefügt.

Die automatische Undo-Verwaltung übernimmt nicht nur die Verwaltung von Anzahl und Größe der Undo-Segmente, sondern kann mit weiteren Raffinessen aufwarten:

- Nach Möglichkeit wird nur eine schreibende Transaktion pro Undo-Segment zugelassen, um die Wahrscheinlichkeit von Zugriffskonflikten auf den Transaktionstabellen im *Header* des Segments zu minimieren.
- Das Überschreiben von Extents ist nicht nur davon abhängig, ob alle schreibenden Transaktionen abgeschlossen wurden, sondern hängt auch von einer »Freigabezeit« ab, die über den Systemparameter `undo_retention` gesteuert wird und festlegt, wie lange *before images* aus beendeten Transaktionen für potenzielle Leser zur Verfügung gehalten werden sollen (Lesekonsistenz und Flashback Query).
- Diese »Freigabezeit« kann jedoch unterschritten werden, wenn der Platz in der Undo-Tablespace knapp zu werden droht (*Space Pressure*).
- Die Segmentverwaltung ist darüber hinaus so flexibel, Extents bei Bedarf zwischen den Segmenten auszutauschen. Auf diese Weise kann ein Segment in der Not, und um entsprechenden Speicherplatz in dem betreffenden Tablespace zu sparen, ein Extent eines fremden Undo-Segmentes »stehlen«, d.h., für sich nutzen.

Die Oracle-Datenbank arbeitet nach einem optimistischen Ansatz, der besagt, dass jede Änderung permanent ist. Daher wird mit der Änderung eines oder mehrerer Datensätze eine Kopie des alten Zustandes in ein Undo-Segment geschrieben, und die neue Information wird als permanenter Datensatz gespeichert, obwohl der Anwender noch keinen `COMMIT`-Befehl ausgeführt hat. Wenn der Anwender mit `ROLLBACK` die Transaktion rückgängig machen möchte, muss also der neue Zustand wieder mit dem alten (*before images*) überschrieben werden.

Neben dieser Möglichkeit des Zurückrollens von Transaktionen erlauben die Undo-Segmente eine Lesekonsistenz von Abfragen. Da für einen gewissen Zeitraum alte und neue Datensätze zur Verfügung stehen können, kann eine Abfrage zeitlich ältere Daten sehen, obwohl diese von einer anderen Transaktion zwischenzeitlich geändert und bereits festgeschrieben wurden.

Im folgenden Beispiel wird durch den Benutzer `u1` eine Transaktion ausgeführt, die den Mitarbeitern der Abteilung 50 fünf Prozent mehr Gehalt zuweist. Es handelt sich um ein Unternehmen mit mehr als 100.000 Mitarbeitern, so dass die Änderung ca. fünf Minuten dauert. Eine Minute nach dem Start der Änderung wird aus der Personalabteilung eine Abfrage gestartet, um die Summe aller Gehälter pro Abteilung zu berechnen. Wenn jetzt auf bereits geänderte und nicht geänderte Daten zugegriffen werden würde, würde dies zu einem falschen Ergebnis führen. Durch das Lesekonsistenz-Modell von Oracle greift die Abfrage automatisch auf das *before image* der bereits geänderten Datensätze zu, um so zu gewährleisten, dass die Information gelesen wird, wie sie zum Start der Abfrage existierte.

```
U1:
UPDATE mitarbeiter
SET gehalt = gehalt * 1,05
WHERE abtnr = 50;
U2:
SELECT abtnr, SUM(gehalt)
FROM mitarbeiter
ORDER BY abtnr;
```

Listing 3.9: Lesekonsistenz-Modell

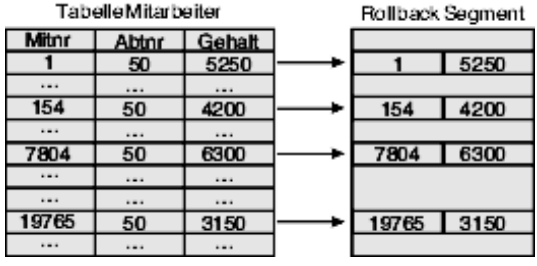

Abbildung 3.2: Consistent Read

Entscheidend ist in diesem Zusammenhang der Serverparameter `undo_retention`. Er gibt eine Zeitdauer an (Standardwert sind 15 Minuten), bis zu der auf *before images* für Lesekonsistenz oder sog. *Flashback Queries* zugegriffen werden kann. Allerdings gilt auch hier, dass bei unzureichend dimensioniertem Undo-Tablespace das Before-Image überschrieben wird, wenn eine DML-Operation den Platz beansprucht, d.h., eine DML-Operation hat Vorrang vor einer eventuellen Leseoperation auf ältere Datensätze. Als Alternative bietet Oracle 10g die Möglichkeit, die Undo-Retention zu garantieren. Da dies den operativen Betrieb jedoch beeinträchtigt, kommt er nur selten zum Einsatz.

Das folgende Bild veranschaulicht das Transaktionsmanagement anhand eines Undo-Segments mit zwei Transaktionen. Die Information der ersten Transaktion T1 kann nach deren Beendigung sofort durch die Transaktion T2 überschrieben werden. Benötigt die Transaktion T2 dann noch weiteren Platz, wird ein zusätzliches Extent in den Ring eingebaut.

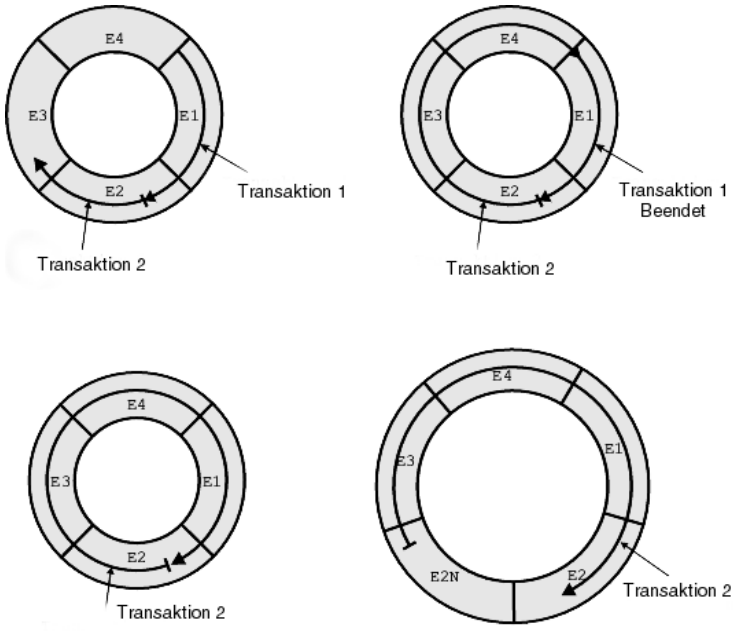

Abbildung 3.3: Undo-Segmente

Die automatische, d.h., systemgesteuerte Undo-Verwaltung erfordert Undo-Tablespaces. Ein Undo-Tablespace kann direkt beim Erstellen der Datenbank angelegt oder nachträglich aufgebaut werden. Im folgenden Beispiel wird der Undo-Tablespace `UNDOTBS` mit einer Größe von 2000 MB und einer automatischen Erweiterung auf maximal 5000 MB aufgebaut:

```
SQL> CREATE UNDO TABLESPACE undotbs
     DATAFILE '/oradata1/PDWH10G/undotbs01.dbf' SIZE 2000M
     AUTOEXTEND ON NEXT 100M MAXSIZE 5000M;
```

Undo-Tablespaces sind stets Locally-Managed Tablespaces, deren Extent-Größen vom System bestimmt werden (`AUTOALLOCATE`). Zusätzliche Klauseln, wie die `DEFAULT STORAGE`- oder `SEGMENT SPACE MANAGEMENT`-Klausel, sind hierbei nicht zulässig und werden mit einer Fehlermeldung quittiert.

`CREATE UNDO TABLESPACE` legt nicht nur einen Tablespace vom Typ UNDO (Attribut `contents` der View `dba_tablespaces` mit Wert UNDO) an, sondern erzeugt in diesem Tablespace auch sofort zehn Undo-Segmente. Die Namen der Undo-Segmente lauten `_SYSSMUxx$`, wobei `xx` eine pro Datenbank fortlaufende Nummer darstellt.

Undo-Tablespaces können darüber hinaus wie andere Tablespaces verwaltet werden: Über den Befehl `ALTER TABLESPACE` lassen sich zusätzliche Dateien anfügen oder der `AUTOEXTEND`-Klausel anpassen. Das Offline-Schalten gelingt allerdings nur, wenn der betreffende Tablespace sowohl nicht mehr aktiv ist als auch keine offenen Transaktionen mehr enthält.

Grundsätzlich können mehrere Undo-Tablespaces pro Datenbank angelegt werden, Oracle arbeitet aber pro Instanz immer nur mit *einem* aktiven Undo-Tablespace. Das Umschalten zwischen vorhandenen Undo-Tablespaces ist grundsätzlich – auch im laufenden Betrieb – möglich, sofern der avisierte Tablespace nicht bereits von einer anderen Instanz aktiviert wurde. In unserem Beispiel wird der Tablespace mit Namen `undotbs02` aktiviert, gleichzeitig sperrt Oracle den alten Undo-Tablespace für neue Transaktionen:

```
SQL> ALTER SYSTEM SET undo_tablespace = undotbs02;
```

Das Umschalten gelingt auch dann, wenn offene Transaktionen existieren. Alte Transaktionen behalten nach wie vor ihren Undo-Kontext, neue Transaktionen werden einem Undo-Segment des neu aktivierten Tablespace zugeordnet.

Achtung: Dies bedeutet auch, dass in RAC(Real Application Clusters)-Umgebungen jede aktive Instanz ihren eigenen Undo-Tablespace benötigt. Es muss hier also mindestens so viele Undo-Tablespaces wie gleichzeitig aktive Instanzen geben. Diese werden zweckmäßigerweise durchnummeriert, z.B. `UNDOTBS01`, `UNDOTBS02` etc.

Undo Management konfigurieren

Die automatische Konfiguration von Undo-Segmenten wird pauschal über den Serverparameter `undo_management` geregelt:

```
undo_management = auto
```

Dieser Parameter ist statisch, d.h., er kann nicht im laufenden Betrieb geändert werden, und er muss für alle Instanzen, die eine Datenbank geöffnet haben, identisch gesetzt sein.

Es versteht sich, dass vor dem Umschalten auf die automatische Undo-Verwaltung ein entsprechender Tablespace angelegt worden sein muss.

Der Wert `manual` schaltet auf die manuelle Konfiguration von Rollbacksegmenten um, die aus älteren Versionen von Oracle bekannt sind. Diese Einstellung ist zwar noch immer der Standardwert, aber nicht mehr zu empfehlen.

Weitere Parameter

Die automatische Undo-Verwaltung bringt einen weiteren Vorteil mit sich: Der Verbleib von Undo-Informationen bereits abgeschlossener Transaktionen muss nun nicht mehr mühsam durch *Try-and-Error*-Methoden ermittelt, sondern kann direkt per Systemparameter eingestellt werden. Im folgenden Beispiel wird festgelegt, dass Undo-Informationen pro Transaktion nach dem `COMMIT` für weitere 3600 Sekunden – also eine Stunde – für lesende Transaktionen verfügbar sind:

```
undo_retention = 3600
```

Achtung! Da Undo-Segmente nicht die gleiche Ringpuffer-Struktur fester Minimalgröße aufweisen wie Rollbacksegmente, zeigt die Erfahrung, dass in Produktivsystemen der Wert für `undo_retention` auf keinen Fall auf dem Standardwert (900 Sekunden) belassen werden sollte. Die hier gezeigte Alternative von einer Stunde ist der grundsätzlich bessere Startwert. Falls man Kenntnis über lang laufende Abfragen oder Transaktionen hat, empfiehlt es sich, mit der `undo_retention` großzügig über deren Laufzeit hinauszugehen.

Der Parameter `undo_retention` ist dynamisch, kann also über den Befehl `ALTER SYSTEM` angepasst werden. Er muss allerdings für alle Instanzen einer Datenbank identisch eingestellt werden. Ein kleiner Wermutstropfen: Der DBA wird hier nicht von allen Überlegungen zur Konfiguration entlastet. Sollte der aktive Undo-Tablespace nicht genügend Platz für neu startende Transaktionen bereithalten und auch nicht dynamisch erweiterbar sein, werden Undo-Einträge abgeschlossener Transaktionen auch bereits vor dem Ablauf dieser Zeitspanne entfernt!

Mit der Option `RETENTION GUARANTEE` wird garantiert, dass für die Dauer der `undo_retention` die alten Daten nicht überschrieben werden, auch wenn dies u.U. zu Fehlern bei der Transaktionsverarbeitung (`ORA-01650 unable to extend rollback segment…`) führen würde.

Den in Oracle9i eingeführten Parameter `undo_suppress_errors`, mit dem Fehlermeldungen wie die unten abgebildete unterdrückt werden konnten, gibt es in Oracle 10g nicht mehr, entsprechende Fehler werden immer ignoriert.

```
SQL> SET TRANSACTION USE ROLLBACK SEGMENT "_SYSSMU13$";
*
FEHLER in Zeile 1:
ORA-30019: Unzulässiger Rollback-Segment-Vorgang in Automatic Undo-Modus
```

Data Dictionary Views

Die aus den Oracle-Versionen 7 und 8 bekannten Views sind auch im Kontext der automatischen Undo-Verwaltung noch relevant:

- `dba_tablespaces` listet die Undo-Tablespaces unter `contents = 'UNDO'`.
- `dba_rollback_segs` listet sowohl systemgenerierte Undo- als auch manuell erstellte Rollback-Segmente. Alle generierten Segmente von nicht aktiven Undo-Tablespaces haben konsequenterweise den Status Offline.
- `dba_segments` führt die systemgenerierten Undo-Segmente unter `segment_type = 'TYPE2 UNDO'`.
- `v$rollstat` und `v$rollname` geben in der gewohnten Weise Auskunft über aktive Undo- und Rollback-Segmente und ihre Statistiken.
- `v$transaction` erlaubt die Zuordnung von Transaktionen zu Undo-Segmenten.
- Seit Oracle9i gibt die View `dba_undo_extents` u.a. die letzte `COMMIT`-Zeit für jedes Extent des Undo-Tablespaces aus.

- Neu hinzugekommen ist auch die View `v$undostat`, über die Statistiken im Zusammenhang mit Undo-Tablespaces ausgegeben werden können. Die View gruppiert die Undo-Nutzung in Intervallen von jeweils zehn Minuten. Für die Optimierung sind vor allem die folgenden Attribute interessant:

  `undoblks` – die Anzahl benutzter Undo-Blöcke

  `txncount` – die Anzahl der Transaktionen

  `maxconcurrency` – die maximale Zahl gleichzeitig operierender Transaktionen

  `unxpblkrelcnt` – Freisetzen von Extents, deren Retention-Periode noch nicht abgelaufen war, für andere Transaktionen. Die angestrebte Retention-Zeit konnte also nicht eingehalten werden.

  `ssolderrcnt` – Anzahl aufgetretener `ORA-01555` Fehler (»snapshot too old«). Hohe Zahlen weisen auf einen zu geringen Wert für `undo_retention` oder (im Zusammentreffen mit hohen Werten der Spalte `unxpblkrelcnt`) auf einen zu kleinen Undo-Tablespace hin.

3.3.6 Read-only-Tablespaces

Für spezielle Operationen kann es sinnvoll sein, einen Tablespace als Read-only zu spezifizieren, so dass nur lesend auf die Daten zugegriffen werden kann, DML-Operationen aber nicht mehr möglich sind. Ein Tablespace kann auch dann in den Read-only-Modus gebracht werden, wenn noch offene Transaktionen vorliegen. Diese offenen Transaktionen werden normal beendet, neue schreibende Transaktionen jedoch nicht mehr zugelassen. Wichtig: Es ist trotz des Read-only-Modus möglich, in dem Tablespace Objekte zu löschen!

Eingesetzt werden Read-only-Tablespaces bei der Benutzung von CD-ROM-Laufwerken z.B. für Bilddateien und bei der Historisierung von Daten, die nicht mehr geändert werden dürfen.

Der wesentliche Unterschied zu Read-Write-Tablespaces besteht darin, dass der Header der zugehörigen Datendateien eingefroren wird, wodurch die Notwendigkeit wiederholter Backups entfällt. Das Schreiben eines Checkpoints wird in diesem Fall ignoriert. Dadurch kann ein Recovery auf einem älteren Stand – einer einzigen Kopie des Read-only-Tablespaces – aufsetzen.

Somit bieten sich Read-only-Tablespaces als Mittel zur Reduzierung des Sicherungsaufkommens an, wenn z.B. Tabellen mit nicht änderbarem Inhalt bzw. »alte Partitionen« von Tabellen in Read-only-Tablespaces ausgelagert werden.

3.3.7 Tablespaces offline setzen

Ein bereits angesprochener Aspekt für die Aufteilung von Objekten auf mehrere Tablespaces ist die Verfügbarkeit. In diesem Punkt ist die Verwendung von Offline-Tablespaces sinnvoll, da hierdurch erreicht wird, dass mit der Datenbank gearbeitet werden kann, obwohl eine Datendatei defekt ist und dadurch wiederhergestellt werden muss. Weitere Anwendungen sind administrative Aufgaben, wie das Sichern im konsistenten Modus oder das Verschieben des Tablespaces auf ein anderes Laufwerk. Folgende Optionen können verwendet werden, um ein Tablespace offline zu setzen:

```
SQL> ALTER TABLESPACE demo OFFLINE NORMAL;
```

Es wird ein Checkpoint geschrieben, um die Konsistenz des Tablespaces zu gewährleisten, anschließend werden alle Datendateien offline gesetzt. Dies ist die Standardoption.

```
SQL> ALTER TABLESPACE demo OFFLINE TEMPORARY;
```

Für die noch geöffneten Datendateien wird ein Checkpoint geschrieben. Sollten alle zugehörigen Datendateien geöffnet sein, so ist das Verhalten identisch zur Option `NORMAL`. Der Befehl wird normalerweise verwendet, wenn eine oder mehrere Datendateien defekt sind und deshalb ein `OFFLINE NORMAL` nicht mehr funktioniert.

```
SQL> ALTER TABLESPACE demo OFFLINE IMMEDIATE;
```

Es wird kein Checkpoint geschrieben, d.h., die Datendateien des Tablespaces sind inkonsistent. Diese Option wird verwendet, wenn es zu einem Fehler in einer Datei kommt, der nur durch ein Media-Recovery behoben werden kann.

```
SQL> ALTER TABLESPACE demo OFFLINE FOR RECOVER;
```

Der Tablespace eines Recovery-Sets wird für Point-in-Time-Recovery offline gesetzt. Ein Recovery-Set bezieht sich hierbei auf eine Anzahl Tablespaces, welche die Konsistenz einer Anwendung gewährleisten können und damit für ein Point-in-Time-Recovery als Einheit betrachtet werden sollen.

3.4 Datenbankblöcke

Die in einem Datenbankmanagementsystem gehaltenen Daten müssen aus Gründen der Wiederherstellbarkeit permanent gespeichert werden, so dass Dateien naturgemäß eine zentrale Rolle für Datenbanken spielen. Die Dateien eines relationalen Datenbankmanagementsystems dürfen aber nicht als »flache Dateien«, d.h. Dateien mit einer simplen Satzstruktur, aufgebaut sein, da Änderungen der Datenstruktur, wie das Hinzufügen von Spalten an bestehende Tabellen, im laufenden Betrieb möglich sein müssen.

Informationen über die Struktur der Daten (so genannte Metadaten) dürfen weiterhin nicht ausschließlich in den verarbeitenden Programmen, sondern müssen bei den Daten gespeichert werden: Dies garantiert die Offenheit der Schnittstellen und die Verwendbarkeit der Daten in unterschiedlichsten Programmen und Entwicklungsumgebungen. Auch um diese Flexibilität zu erreichen, verwendet der Oracle Server eine Blockstruktur als Grundlage für seine Dateistrukturen.

Die kleinste Bearbeitungseinheit für fast alle Komponenten des Oracle Servers ist demnach ein Datenbankblock. Ob es sich um I/O-Problematiken, physikalische Zugriffskonflikte oder die Entscheidungsstrategien des Optimizers handelt, der Block spielt eine zentrale Rolle. Weiterhin ist zu erwähnen, dass alle Datenbankinhalte, also auch das Data Dictionary, das die Metadaten enthält, in der Blockstruktur gespeichert sind.

Die Größe eines Datenbankblocks ist keinesfalls a priori festgelegt; sie kann vielmehr beim Anlegen der Datenbank den Gegebenheiten (Anforderungen, Anwendung, Hardware und Betriebssystem) angepasst werden. Sie ist auf jeden Fall ein

Vielfaches von 1 KB, gängige Werte sind 4, 8 oder 16 KB. Wenn die Datenbank einmal angelegt ist, ist die Datenbankblockgröße nicht mehr änderbar und gilt für den SYSTEM-Tablespace und als Standardwert für alle weiteren Tablespaces. Innerhalb einer Datenbank können jedoch mehrere Blockgrößen verwendet werden, jedoch nicht innerhalb einer Datendatei oder eines Tablespaces. Für die Dateien des Datenbanksystems, die Daten enthalten, gilt, dass ihre Dateigröße ein ganzzahliges Vielfaches ihrer Blockgröße ist.

Ein Datenbankblock besteht immer aus zwei logischen Teilen: dem Kopf und dem Inhalt. Beide Teile sind variabel in der Größe, so dass möglichst keine vorgefertigten Beschränkungen wie z.B. Satzzahl pro Block vorhanden sind. Damit sich die beiden Teile beim Wachstum nicht behindern, befindet sich der Kopf am Anfang des Blocks und wächst zum Ende hin; der eigentliche Inhalt beginnt hingegen am Blockende und wächst zum Anfang hin.

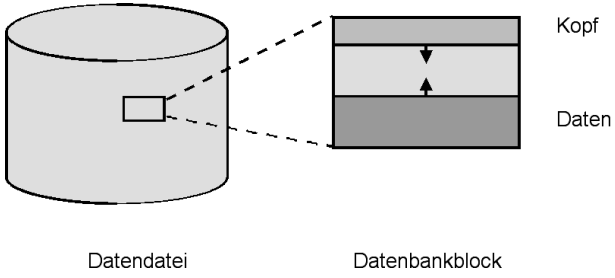

Abbildung 3.4: Datendatei und Datenbankblock

3.5 Online-Redolog-Dateien

Die Redolog-Dateien enthalten ein physikalisches Transaktionsprotokoll. Dies ist nicht zu verwechseln mit einem logischen Transaktionsprotokoll, das eine Sequenz von SQL-Kommandos enthalten würde. Alle Änderungen an Datenbankblöcken werden hier protokolliert. Die Protokollierung hat den einfachen Sinn, dass in einem Fehlerfall die logischen Änderungen an der Datenbank durch das nochmalige Durchführen der Blockänderungen wiederhergestellt werden können. Die Eigenschaft der Wiederherstellbarkeit ist eine rudimentäre Anforderung an ein Datenbankmanagementsystem – ohne sie wäre ein produktiver Einsatz nicht möglich.

Redolog-Dateien speichern also *redo*-Informationen des Oracle-Systems, d.h. Informationen, die zum Wiederherstellen (*redo*) von Blockzuständen notwendig sind – im Gegensatz zu den Undo-Segmenten, die Informationen zum Zurücksetzen (*undo*) von Blockzuständen speichern. Sie beinhalten ein Transaktionsprotokoll der Datenbankaktivitäten, das im Fehlerfall für die Wiederherstellung der Datenbankkonsistenz nach einem Absturz der Instanz benötigt wird. Als solches enthalten sie Daten zu allen möglichen Blöcken (Daten-, Index- und auch Undo-Blöcken). Notwendig wird das Transaktionsprotokoll durch die asynchrone Arbeitsweise des Database-Writer-Prozesses, der nur in Intervallen oder bei Platzmangel Blöcke aus dem Datenbank-Puffer auf die Platte schreibt. Damit wäre also z.B. bei einem Stromausfall ein Transaktionsverlust vorprogrammiert.

Tritt bei einer Oracle-Instanz durch einen Stromausfall, einen Hardware- oder Softwarefehler ein Abbruch auf, so werden bei einem Wiederanlauf automatisch, durch den System-Monitor(SMON)-Prozess angestoßen, die Transaktionen aus den benötigten Redolog-Dateien nachgefahren.

Eine Oracle-Datenbank kennt zwei Typen von Redolog-Dateien: die vom RDBMS ständig zyklisch beschriebenen Online-Redolog-Dateien sowie die (weiter unten beschriebenen) Offline- oder archivierten Redolog-Dateien, die optional (im so genannten *Archivelog*-Modus) nach dem Schließen von Online-Redolog-Dateien als eine chronologische Folge von durchnummerierten Redolog-Dateikopien entstehen.

3.5.1 Konfiguration

Bei der Konfiguration von Redolog-Dateien werden die Bereiche Sicherheit und Performance des Oracle-Systems tangiert. Diese Ziele stehen zunächst im Widerspruch, können aber durch Spiegelung der Redolog-Dateien gut in Einklang gebracht werden.

Lage der Redolog-Dateien

Für transaktionsorientierte Oracle-Systeme spielt der Durchsatz des Logwriter-Prozesses eine entscheidende Rolle. Bei einem `COMMIT`-Befehl gilt die Transaktion so lange als nicht beendet, bis alle notwendigen Protokollinformationen in die Redolog-Dateien geschrieben worden sind. Aus Gründen der Performance sollten also die Redolog-Dateien auf schnelle Platten gelegt werden, auf denen ansonsten wenige Aktivitäten stattfinden. Bei extrem hohen Anforderungen sollte auch über Plattenstriping für die Redolog-Dateien nachgedacht werden.

Die Trennung der Redolog-Dateien von den Datendateien empfiehlt sich ohnehin aus Sicherheitsaspekten, da ja die Redolog-Dateien im Fehlerfall, bei einem Platten-Crash, zur Wiederherstellung der Datendateien benötigt werden.

Anzahl der Redolog-Dateien

Eine Oracle-Datenbank benötigt mindestens zwei Gruppen von Redolog-Dateien, zwischen denen der Logwriter-Prozess umschalten kann. Mehr als zwei Redolog-Gruppen zu konfigurieren, kann folgende Gründe haben:

1. Wartezustände auf Checkpoints

   Eine Redolog-Datei kann erst dann überschrieben werden, wenn sämtliche Blöcke aus dem Database Buffer Cache in der SGA vom Database Writer in die Datendateien geschrieben worden sind, für deren Wiederherstellung bei Ausfall der Instanz die Informationen aus der jeweiligen Redolog-Datei gebraucht werden. Dieser Konsistenzabgleich ist als *Checkpoint* bekannt. Dauert jetzt bei einem sehr großen Database Buffer Cache und einer hohen Transaktionslast der Checkpoint länger als das Vollschreiben der zweiten Gruppe, ist ein Wartezustand die Folge. In der Alert-Datei der Instanz wird die Meldung *Checkpoint not complete* protokolliert.

Da der Checkpoint-Algorithmus die Parallelisierung von Checkpoints erlaubt, kann dieser Wartezustand durch eine höhere Anzahl von Redolog-Gruppen beseitigt werden.

2. Performance des Archivierungsprozesses

   Die Archivierungsprozesse ARCn sichern voll geschriebene Redolog-Dateien in ein dafür konfiguriertes Verzeichnis, idealerweise auf einer eigenen Platte bzw. einem eigenen Volume. Ist der ARCn-Prozess damit noch nicht fertig, wenn die Datei wieder überschrieben werden soll, so erzeugt das einen Wartezustand. Das kann gerade bei hoher Transaktionslast auftreten. Diese Lastspitzen können über eine größere Anzahl von Redolog-Gruppen abgefangen werden, da es dann entsprechend länger dauert, bis die betreffende Datei wieder überschrieben werden soll, und der ARCH-Prozess damit mehr Zeit zur Verfügung hat, diese zu sichern.

Größe der Redolog-Dateien

Die Größe der Redolog-Dateien wird aus Performance-Gründen verändert. Immer wenn auf die nächste Redolog-Gruppe umgeschaltet wird, weil die Datei der aktuellen Gruppe zu 100 Prozent gefüllt ist, wird ein Checkpoint initiiert. Während des Checkpoints müssen alle modifizierten Blöcke aus der SGA durch den DBWn-Prozess in die Datendateien geschrieben werden, was bei einer großen SGA bei hoher Transaktionslast zu einer hohen Systemlast führen kann. Eine Reduzierung der Checkpoints führt also zu einer Reduzierung der Systemlast. Werden also die Redolog-Dateien vergrößert, verlängert sich die Zeit zwischen zwei Checkpoints, und die Systemlast sinkt.

Allerdings verlängert sich die Zeit beim Wiederanlauf des Oracle-Systems, da z.B. nach einem Stromausfall mehr Transaktionen seit dem letzten Konsistenzzeitpunkt nachgefahren werden müssen. Jedoch ist diese Zeit in der Praxis zu vernachlässigen. Zu dem Wiederanlauf des Rechners kommen dann erfahrungsgemäß einige Minuten Instance-Recovery-Zeit. Die genau benötigte Zeit lässt sich schwer voraussagen; sie ist einerseits abhängig von den Leistungsdaten des Systems, andererseits von der Art der protokollierten Aktionen. In den allermeisten Fällen erweist sich diese Zeit nicht als kritisch.

Bei einem Verlust einer Redolog-Gruppe wäre allerdings die Menge der verlorenen Transaktionen größer. Also stellen größere Redolog-Dateien ein höheres Risiko dar.

Dieses Risiko kann dadurch minimiert werden, dass man die Redolog-Dateien spiegelt, also Gruppen mit zwei oder drei Mitgliedern (*Members*) konfiguriert oder die Redolog-Dateien auf gespiegelte Platten legt. Für Systeme mit hoher Transaktionslast sind Redolog-Dateien mit einer Größe von mindestens 100 MB (was mittlerweile dem Standardwert, zumindest für Oracle Managed Files, entspricht) empfehlenswert, bei großen Systemen werden durchaus 500 MB und mehr konfiguriert.

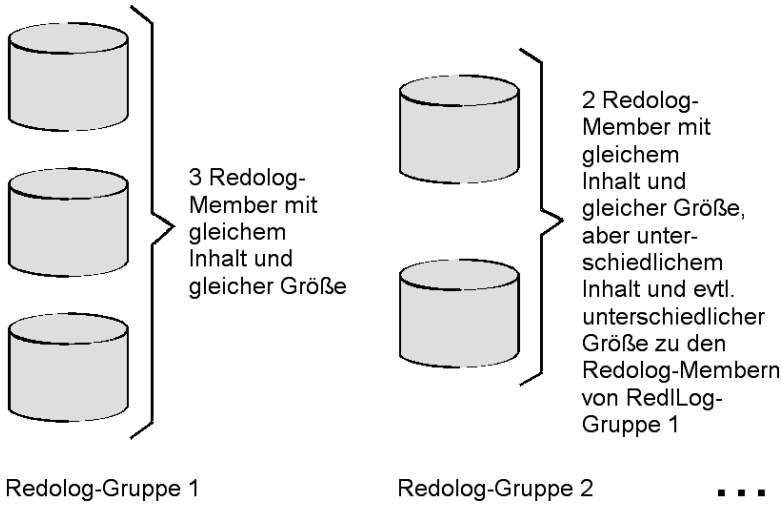

Abbildung 3.5: Redolog-Gruppen und Redolog-Member

Alle Redolog-Gruppen sollten identische Dateigrößen aufweisen. Auch wenn die Datenbank prinzipiell unterschiedliche Redolog-Größen zulässt, führt dies in der Praxis nur zu Problemen und sollte daher vermieden werden.

Abhängige Serverparameter

Zur Beeinflussung des Checkpoint-Verhaltens der Oracle Instanz gibt es diverse Parameter, die als historisch gewachsene Strukturen betrachtet werden können:

- `log_checkpoint_interval` und `log_checkpoint_timeout` sind die auch schon in früheren Oracle-Versionen eingesetzten Parameter, die harte Checkpoints innerhalb der Befüllung einer Redolog-Datei ausgelöst haben. Die Werte werden für `log_checkpoint_interval` in 512-B-Blöcken, für `log_checkpoint_timeout` in Sekunden angegeben.

  Eine grundsätzliche Konfigurationsempfehlung war früher, Checkpoints innerhalb von Redolog-Dateien zu vermeiden und somit Anzahl und Häufigkeit von Checkpoints durch die Größe der Redolog-Dateien zu steuern. Hierzu setzte man `log_checkpoint_interval` größer als die Redolog-Datei-Größe und `log_checkpoint_timeout` auf den Wert 0.

  Die aktuelle Empfehlung lautet, die Parameter auf den Wert 0 zu setzen oder nicht zu setzen, womit ihre Funktion ausgeschaltet ist.

- Mit Oracle8i wurde für die Enterprise Edition der Parameter `fast_start_io_target` eingeführt, der die Anzahl der notwendigen IOs für eine Instanzwiederherstellung beschränken sollte. Der Parameter ist zwar auch noch in Oracle 10g vorhanden, wird aber grundsätzlich nicht mehr empfohlen.

- Die Empfehlung gilt aktuell für den mit Oracle9i eingeführten Parameter `fast_start_mttr_target`, der eine Vorgabe bezüglich der erlaubten Zeit für eine Instanzwiederherstellung in Sekunden erlaubt (auch hier gilt: für die Enterprise Edition). Wie der zuvor genannte Parameter wird hiermit *Fast Start Checkpointing* aktiviert. Das bedeutet, dass der Database Writer so viel zusätzliche Schreiblast bekommt, um den nächsten Checkpoint bereits im Vorfeld so weit voranzutreiben, dass die angegebene Zeit für die Instanzwiederherstellung ausreichend ist.

  In Oracle 10g wurde der zugrunde liegende Algorithmus insofern verbessert, dass Zeiten, in denen IO-Kapazitäten frei sind, bevorzugt für diesen Zweck genutzt werden.

  Für Systeme mit definierten Anforderungen an Wiederanlaufzeiten im Fehlerfall stellt die Möglichkeit der Zeitvorgabe eine komfortable Realisierung der Anforderungen dar.

3.5.2 Administrationskommandos

Im Folgenden werden die wichtigen Kommandos für die Administration der Redolog-Dateien beispielhaft vorgestellt. Für die komplette Syntax sei auf das Handbuch *Oracle 10g SQL Reference* verwiesen. Die hier vorgestellten Kommandos gelten genauso für Oracle9i und Oracle8i.

Zusätzlich können Redolog-Dateien über Oracle Managed Files verwaltet werden. Dabei werden dann die Dateinamen und ev. Größen automatisch vergeben (siehe Kapitel 3.9).

Anlegen einer neuen Redolog-Gruppe

```
SQL> ALTER DATABASE ADD LOGFILE GROUP 3
     '/disk1/verzeichnis/redo_3_1.log' SIZE 50M;
```

Der Name der Redolog-Datei sollte die Gruppennummer (hier 3) und die Nummer des Mitglieds einer Gruppe beinhalten (hier 1). Die Dateigröße als Eigenschaft der Redolog-Gruppe sollte angegeben werden, um nicht auf versionsspezifischen Standardwerten zu landen.

Hinzufügen eines Mitglieds zu einer Redolog-Gruppe

```
SQL> ALTER DATABASE ADD LOGFILE MEMBER
     '/disk1/verzeichnis/redo_3_2.log' TO GROUP 3;
```

Umbenennen von Redolog-Dateien

```
SQL> ALTER DATABASE
     RENAME FILE '/disk1/verzeichnis/redo_1_1.log'
     TO '/disk2/verzeichnis_neu/redo_1_1.log';
```

Hiermit wird nicht etwa die Redolog-Datei selbst verschoben, sondern lediglich ihr Name in der Kontrolldatei geändert. Voraussetzungen für dieses Kommando sind, dass die Gruppe nicht gerade im Zugriff des Logwriter-Prozesses ist, die Quelldatei identisch mit den Informationen aus `v$logfile` ist und die Zieldatei existiert, also vorher auf Betriebssystemebene kopiert worden ist.

Löschen eines Mitglieds einer Redolog-Gruppe

```
SQL> ALTER DATABASE DROP LOGFILE MEMBER
     '/disk1/verzeichnis/redo_3_2.log';
```

Dieser Befehl löscht den Eintrag aus der Kontrolldatei. Auf Betriebssystemebene muss die Datei zusätzlich gelöscht werden. Bei Oracle Managed Files werden die Dateien automatisch von der Platte gelöscht.

Löschen einer Redolog-Gruppe

```
SQL> ALTER DATABASE DROP LOGFILE GROUP 3;
```

Voraussetzungen sind, dass die jeweilige Gruppe nicht im Zugriff des Logwriter-Prozesses ist und mindestens zwei Gruppen verbleiben. Auf Betriebssystemebene muss die Datei zusätzlich gelöscht werden. Bei Oracle Managed Files werden die Dateien automatisch von der Platte gelöscht.

Initialisieren einer Gruppe

```
SQL> ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP 1
     UNRECOVERABLE;
```

Mit diesem Befehl kann eine Redolog-Gruppe oder auch eine einzelne Redolog-Datei initialisiert werden. Die Wirkung entspricht dem Löschen und Neuanlegen der Gruppe, jedoch funktioniert dieser Befehl auch dann, wenn nur zwei Redolog-Gruppen angelegt wurden. Der Status der Gruppe ist anschließend `UNUSED`, die *Log Sequence Number* wird auf 0 zurückgesetzt. Der Zusatz `UNARCHIVED` erlaubt sogar die Initialisierung, auch wenn die Datenbank im Archivierungsmodus läuft und die Redolog-Datei noch nicht archiviert ist. Die Option `UNRECOVERABLE` bewirkt, dass eine Initialisierung stattfindet, obwohl Inhalte der Redolog-Datei noch für eine Wiederherstellung der Datenbank benötigt werden. Diese Optionen sollten jedoch nur im äußersten Notfall verwendet werden.

3.5.3 Überwachung von Redolog-Dateien

Zur Überwachung der Redolog-Dateien stehen folgende V$-Tabellen zur Verfügung:

- `v$log`

  Die Information listet unter anderem die Größe der Dateien (`BYTES`), den Status der Archivierung (`ARCHIVED`) und den Status der Datei (`STATUS`) auf. Folgende Status sind dabei möglich:

  - `INACTIVE`: wird zurzeit nicht benutzt, kann also gelöscht werden, solange mehr als zwei Gruppen übrig bleiben.
  - `ACTIVE`: Die Datei wird für ein Instance-Recovery benötigt, kann also nicht gelöscht werden.
  - `UNUSED`: Die Datei wurde noch nie benutzt, z.B. nach dem neuen Anlegen.
  - `CURRENT`: in dieser Datei wird zurzeit geschrieben.

- `v$logfile`

  Neben der Gruppenzugehörigkeit werden in dieser Tabelle auch der Status und die Namen der Redolog-Dateien angezeigt, unter ihnen auch Oracle Managed Files.

  Um sich die Größe und Lage der Redolog-Dateien anzusehen, muss man einen Join über die Tabellen `v$log` und `v$logfile` mit der Spalte `GROUP#` durchführen, z.B. in der Form:

```
SQL> SELECT v.group#, vl.member, v.bytes, v.status
     FROM   v$log v,v$logfile vl
     WHERE  v.group# = vl.group#;
   GROUP# MEMBER                                    BYTES STATUS
---------- ---------------------------------- ---------- ----------------
         1 D:\ORACLE\ORADATA\QJA10G\REDO01.LOG   10485760 INACTIVE
         2 D:\ORACLE\ORADATA\QJA10G\REDO02.LOG   10485760 CURRENT
         3 D:\ORACLE\ORADATA\QJA10G\REDO03.LOG   10485760
INACTIVE
```

Listing 3.10: Redolog-Informationen

- `v$loghist` und `v$log_history`

  In diesen V$-Tabellen werden Informationen über archivierte Redolog-Dateien wie Namen, Zeitstempel und SCN-Nummer verwaltet. Diese Auszüge kommen aus der Redolog-Historie in den Kontrolldateien.

  Zusätzlich werden sämtliche Redolog-Switches mit Zeitstempel in der Alert-Datei einer Instanz protokolliert. Hier sind auch die Wartezustände vermerkt, die durch Checkpoints oder die Archivierung bedingt sind, meist mit Zeitangaben.

3.5.4 Überlegungen zur Spiegelung von Redolog-Dateien

Redolog-Dateien sind die kritischen Dateien einer Oracle-Datenbank im Fehlerfall. Datendateien und Kontrolldateien können wiederhergestellt werden, vorausgesetzt, es existiert eine Sicherung. Redolog-Dateien beinhalten aktuelle Transaktionsinformationen, die insbesondere nach einer Beschädigung an einer Festplatte zur Wiederherstellung der Datendateien benötigt werden. Der Verlust einer kompletten Redolog-Gruppe bedeutet Datenverlust! Aus diesem Grunde ist es für Produktionsdatenbanken erforderlich, die Redolog-Dateien zu spiegeln.

Dabei steht zur Auswahl, entweder eine Oracle-seitige Spiegelung einzuschalten (also pro Redolog-Gruppe mindestens zwei Member auf verschiedenen Platten zu konfigurieren) oder die Dateien der Redolog-Gruppen auf betriebssystem- oder hardwareseitig gespiegelte Platten zu legen. Von einer Spiegelung über RAID 5 sollte abgesehen werden, da die benötigte IO-Kapazität durch solche Systeme i.A. nicht gewährleistet werden kann.

Fällt ein Spiegel aus, wird die Oracle-Instanz weiterlaufen, der Logwriter-Prozess schreibt in die verbleibende Datei der Gruppe. Bei der Oracle-seitigen Spiegelung hat man den Vorteil, dass der Schreibfehler auf das fehlende Mitglied in der Alert-Datei protokolliert wird. Betriebssystem- oder hardwareseitige Spiegelungen sind für den Logwriter-Prozess transparent.

3.6 Kontrolldateien

Eine zentrale Aufgabe haben Kontrolldateien: Sie enthalten neben einigen Zeitstempeln und Konsistenzinformationen die physikalische Struktur der Oracle-Datenbank, d.h. die Namen und Größen aller Daten- und Online-Redolog-Dateien. Für eine Datenbank gibt es zu jedem Zeitpunkt genau eine gültige Version der Kontrolldatei; diese kann gespiegelt betrieben werden.

Die Kontrolldateien sind die kleinsten Komponenten (wenige MB) einer Oracle-Datenbank, haben aber sowohl für die Verfügbarkeit als auch für die Konsistenz der Datenbank eine zentrale Bedeutung, besonders im Wiederherstellungsfall.

In einer Kontrolldatei sind Informationen über die Datenbank und alle Dateien abgelegt, die zur Datenbank gehören. Dabei besteht eine Eins-zu-eins-Zuordnung der Kontrolldatei zu einer Datenbank.

Wichtige Informationen, die in einer Kontrolldatei gespeichert werden, sind unter anderem:

- der Name und Erstellungszeitpunkt der Datenbank (`v$database`)
- die Namen der zur Datenbank gehörenden Kontroll-, Daten-, Temporär- und Redolog-Dateien mit kompletter Pfadangabe und Statusinformationen wie Offline, Read-only usw. (`v$controlfile`, `v$datafile`, `v$tempfile`, `v$logfile`, `v$log`)
- Informationen über Tablespaces (`v$tablespace`)
- Informationen über den Archivierungsstatus (`v$archived_logs`)
- Redolog-Historie (`v$log_history`)
- Informationen für den Recovery-Manager (`v$backup_<xxx>`)
- die aktuelle Log-Sequence-Nummer
- Checkpoint-Informationen

Die Tatsache, dass die Informationen aus den o.a. v$-Views aus der Kontrolldatei kommen, kann leicht überprüft werden, indem man sie im MOUNT-Status einer Instanz abfragt.

Die Kontrolldateien sind sowohl beim Starten des Oracle-Systems als auch während des normalen Datenbankbetriebs im Zugriff. Die Aktualisierung der Kontrolldateien erfolgt über die Instanz beim Auftreten eines Checkpoints oder bei Strukturänderungen an der Datenbank. Strukturänderungen der Datenbank sind beispielsweise das Hinzufügen, Umbenennen und Löschen von Datendateien oder Tablespaces, Redolog-Gruppen oder Redolog-Membern.

Konsistenzinformationen bezüglich der Datendateien werden redundant in der Kontrolldatei und in den *Header* der betreffenden Dateien gespeichert und bei Bedarf – z.B. beim Starten der Datenbank – verglichen. Ein Abgleich der Informationen findet darüber hinaus bei einem Checkpoint statt. Zudem speichert der Checkpoint-Prozess in Intervallen von drei Sekunden den Status der Redolog-Dateien in der Kontrolldatei, um parametrierte Grenzen wie z.B. `log_checkpoint_interval` oder `log_checkpoint_timeout` abgleichen zu können.

Diese Konsistenzinformationen bestimmen bei einer Instanzwiederherstellung oder bei einer Medienwiederherstellung, welche Informationen aus Redolog-Dateien oder archivierten Redolog-Dateien nachgefahren werden müssen, um die Datenbank wieder auf einen konsistenten Zustand zu bringen. Sind diese Informationen nicht vollständig vorhanden, muss eine unvollständige Medienwiederherstellung durchgeführt werden, was einen Datenverlust nach sich zieht.

Die Kontrolldateien sind somit auch ein wichtiger Bestandteil der Sicherung einer Oracle-Datenbank. Wenn Strukturänderungen wie das Hinzufügen eines neuen Tablespaces oder einer neuen Daten- oder Redolog-Datei durchgeführt werden, sollte anschließend sofort eine Sicherung der Kontrolldatei vorgenommen werden.

3.6.1 Administration der Kontrolldateien

Anlegen der Kontrolldateien

Die Kontrolldatei wird automatisch beim Anlegen der Datenbank erzeugt. Die Lage der Kontrolldatei wird dabei über den Serverparameter `control_files` bestimmt. Die Größe wird abhängig vom `CREATE DATABASE`-Kommando der spezifizierten Obergrenzen errechnet. Die dafür relevanten Parameter sind `MAXINSTANCES`, `MAXLOGFILES`, `MAXLOGMEMBERS`, `MAXLOGHISTORY` und `MAXDATAFILES`.

Hier ist anzuraten, die Parameter mit Reserven zu versehen. Ist die hier definierte Obergrenze überschritten, so können keine weiteren Dateien des entsprechenden Typs an die Datenbank angehängt werden. Dieses Problem ist dann nur über das Anlegen einer neuen Kontrolldatei zu lösen, was man in der Praxis jedoch vermeiden sollte.

Spiegelung der Kontrolldateien

Die Wichtigkeit der Kontrolldateien, sowohl für den laufenden Betrieb als auch für den Wiederanlauf des Oracle-Systems, insbesondere bei der Wiederherstellung der Datenbank, ist im vorangehenden Abschnitt beschrieben worden. Aus diesem Grunde ist es sehr empfehlenswert, die Kontrolldateien zu spiegeln. Auch hier stehen datenbank-, betriebssystem- und hardwareseitige Spiegelung zur Auswahl.

Im Folgenden wird das Verfahren zur Einrichtung einer datenbankseitigen Spiegelung beschrieben (Pfadangaben für Unix):

Der aktuelle Eintrag in der Parameterdatei laute beispielsweise:

```
control_files = '/oradata1/PDWH10G/control01.ctl'
```

Zunächst wird der Eintrag geändert auf:

```
control_files = '/oradata1/PDWH10G/control01.ctl',
                '/oradata2/PDWH10G/control02.ctl'
```

Dies geschieht entweder manuell bei Verwendung einer herkömmlichen Parameterdatei oder – bei Verwendung einer Serverparameterdatei – über folgendes Kommando:

```
SQL> ALTER SYSTEM SET CONTROL_FILES =
     '/oradata1/PDWH10G/control01.ctl',
     '/oradata2/PDWH10G/control02.ctl'
     SCOPE = SPFILE;
```

Anschließend wird die Instanz heruntergefahren und die Kontrolldatei auf Betriebssystemebene kopiert:

```
$ cp /oradata1/PDWH10G/control01.ctl
     /oradata2/PDWH10G/control02.ctl
```

Zum Schluss wird die Instanz wieder hochgefahren.

Es ist unbedingt zu beachten, dass die Kontrolldateien nur bei gestoppter Instanz kopiert werden dürfen, da ansonsten die Versionsstände der Dateien differieren und zu einem Fehler beim Starten des Oracle-Systems führen.

Umbenennen oder Verlagern der Kontrolldateien

Für das Umbenennen oder Verlagern der Kontrolldateien gilt das gleiche Verfahren wie bei der Erstellung einer Spiegelung. Auch hier ist unbedingt die Reihenfolge des Stoppens, der Umbenennung und des Startens einzuhalten.

Sichern und Erzeugen von Kontrolldateien

Eine Kontrolldatei kann während des laufenden Betriebes durch folgendes Kommando gesichert werden:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE
     TO '/orabackup/PDWH10G/control_bck.ctl';
```

Bei Verlust der Kontrolldatei einer Datenbank kann diese durch ein `CREATE CONTROLFILE`-Kommando wieder aufgebaut werden. Im Kommando müssen alle Dateien angegeben werden, die zur Datenbank gehören, sowohl alle Redolog- als auch alle Datendateien. Da bei der Erstellung eines solchen Skripts Fehler vorprogrammiert sind, existiert die Möglichkeit des automatischen Generierens.

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

Die daraus resultierende Trace-Datei wird in das Verzeichnis geschrieben, das als `user_dump_dest` definiert ist. Die Trace-Datei enthält das Startup-Kommando und das `CREATE CONTROLFILE`-Kommando, inklusive Kommentierung der einzelnen Schritte.

Dieses Skript kann auch dann genutzt werden, wenn durch zu klein gewählte Einstellungen beim Anlegen der Datenbank – die Klauseln `MAXDATAFILES` und `MAXLOGFILES` – strukturelle Erweiterungen nicht mehr möglich sind. Das Skript kann editiert werden, wobei man dann höhere `MAX`-Werte einträgt.

Das Erzeugen einer Kontrolldatei ist jedoch immer ein Vorgang, der große Sorgfalt erfordert und für den das Oracle-System gestoppt werden muss. In der Oracle-Dokumentation wird angeraten, sowohl vor als auch nach der Ausführung des `CREATE CONTROLFILE`-Kommandos eine Sicherung zu machen.

Beispiel eines `CREATE CONTROLFILE`-Kommandos:

```
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "PDWH10G" NORESETLOGS
ARCHIVELOG
   MAXLOGFILES 32
   MAXLOGMEMBERS 3
   MAXDATAFILES 254
   MAXINSTANCES 1
   MAXLOGHISTORY 454
LOGFILE
   GROUP 1 '/oradata1/PDWH10G/redo_1_1.log' SIZE 50M,
   GROUP 2 '/oradata1/PDWH10G/redo_2_1.log' SIZE 50M,
   GROUP 3 '/oradata1/PDWH10G/redo_3_1.log' SIZE 50M
DATAFILE
   '/oradata1/PDWH10G/system01.dbf',
   '/oradata1/PDWH10G/sysaux01.dbf',
   '/oradata1/PDWH10G/undotbs01.dbf',
   '/oradata1/PDWH10G/users01.dbf',
   '/oradata1/PDWH10G/tools01.dbf',
   '/oradata1/PDWH10G/indx01.dbf'
CHARACTER SET WE8ISO8859P15 ;
```

Kontrolldateien und Oracle Managed Files

Die Kontrolldateien können auch als Oracle Managed Files angelegt werden. Das bedeutet, dass nicht durch den Serverparameter `control_files` die Namen bestimmt werden, sondern dass das Oracle-System beim Anlegen der Datenbank eigene eindeutige Dateinamen generiert. Die Lage der Kontrolldateien wird dann über den Serverparameter

```
db_create_online_log_dest_n = /platte/pfad
```

bestimmt. Dieser Parameter bestimmt darüber hinaus die Lage der Online-Redolog-Dateien. Für die Spiegelung müssen dann entsprechend zwei oder mehr Destinationen angegeben werden.

Bei Verwendung einer Parameterdatei alten Stils (`init.ora`-Datei) müssen die generierten Dateinamen über die View `v$controlfile` ermittelt und anschließend über den Parameter `control_files` eingetragen werden. Bei Verwendung einer Serverparameterdatei entfällt diese Maßnahme.

3.7 Parameterdateien

Die Parameterdatei enthält sämtliche Serverparameter für einen Oracle-Server. Hieraus werden beim Systemstart die Konfiguration und das Verhalten der Instanz bestimmt. Unter anderem enthält die Parameterdatei den oder die Namen der Kontrolldateien. Seit Oracle9i gibt es sie in zwei Varianten.

Herkömmliche Parameterdatei (PFILE bzw. init.ora)

Zum Begriff Parameterdatei sind zwei Anmerkungen zu machen:

1. Der tatsächliche Name der Parameterdatei ist in den meisten Fällen anders als `init.ora`. Meist ist der Name der Oracle-Instanz (zum Begriff der Instanz siehe weiter oben) enthalten wie in `initPDWH10G.ora`; er kann aber auch frei gewählt werden. Beim Hochstarten der Datenbanken wird – wenn nichts anderes angegeben wird – standardmäßig die Datei mit dem Namen `init<ORACLE_SID>.ora` im Verzeichnis `<ORACLE_HOME>\database` (für MS-Windows) bzw. `<ORACLE_HOME>/dbs` (für andere Plattformen) benutzt, daher findet man die Parameterdatei häufig unter diesem Pfad.
2. Die Start- oder Serverparameter einer Oracle-Instanz heißen in der Oracle-Anwendergemeinde »`init.ora`-Parameter«, da sie traditionell in der »`init.ora`-Datei« gespeichert werden. Da die Speicherung der Parameter in der `init.ora`-Datei nur noch eine von vielen Möglichkeiten ist, wird in diesem Buch die Bezeichnung Serverparameter vorgezogen.

Serverparameterdatei

Die Konfiguration einer Instanz über eine herkömmliche Parameterdatei ist problematisch, wenn Start- und Stoppvorgänge in einer verteilten Rechnerumgebung vorgenommen werden. Das liegt daran, dass das Client-Werkzeug, das den `STARTUP`-Befehl ausführt, die Parameterdatei liest. Da die Datei jedoch normalerweise auf dem Server gespeichert ist, muss diese vom Client entweder über das Netzwerk gelesen werden oder eine Kopie auf dem Client existieren. Beide Möglichkeiten sind fehleranfällig.

Seit Oracle9i besteht die Möglichkeit, alternativ zur herkömmlichen Parameterdatei eine so genannte *Serverparameterdatei* zu nutzen. Diese wird in einem Binärformat auf dem Server gespeichert und kann aus einer bestehenden herkömmlichen Parameterdatei erzeugt werden. Falls eine Serverparameterdatei existiert, wird diese anstelle der herkömmlichen Parameterdatei zur Konfiguration beim Instanzstart benutzt. Der Inhalt der Serverparameterdatei wird mit Hilfe von `ALTER SYSTEM`-Befehlen modifiziert.

Die Serverparameterdatei wird beim `STARTUP`-Kommando stets auf der Serverseite im Verzeichnis `$ORACLE_HOME/dbs` (auf Unix) bzw. `%ORACLE_HOME%\database` (auf MS-Windows) unter dem Namen `spfile<ORACLE_SID>.ora` gesucht.

Achtung: Auch wenn eine Serverparameterdatei auf den ersten Blick bequem im Texteditor manuell editiert werden kann, stellt sich auf den zweiten Blick heraus, dass sie Prüfsummen enthält. Nach einer manuellen Veränderung ist die Serverparameterdatei daher meistens unbrauchbar und wird beim Versuch, die Instanz zu starten, abgelehnt.

Stattdessen kann über die Kommandos

```
SQL> CREATE SPFILE='/pfad/auf/server/spfileXYZ.ora'
     FROM   PFILE ='/pfad/auf/client/initXYZ.ora';
```

bzw.

```
SQL> CREATE PFILE ='/pfad/auf/client/initXYZ.ora'
     FROM   SPFILE='/pfad/auf/server/spfileXYZ.ora';
```

zwischen einer herkömmlichen Parameterdatei und einer Serverparameterdatei umgewandelt werden.

Am einfachsten kann dieser Befehl mit den Standardvorgaben verwendet werden. Der Befehl

```
SQL> CREATE SPFILE FROM PFILE;
```

erstellt die Datei `spfile<ORACLE_SID>.ora` im Standardverzeichnis (`$ORACLE_HOME/dbs` bzw. `%ORACLE_HOME%\database`), wenn diese noch nicht existiert bzw. wenn die Datenbank nicht mit dieser Datei hochgefahren wurde. Ansonsten wird der Befehl abgelehnt.

```
SQL> CREATE PFILE FROM SPFILE;
```

Hiermit wird ebenfalls im Standardverzeichnis die Datei `init<ORACLE_SID>.ora` erstellt. Allerdings sollten Sie hierbei aufpassen, da eine ältere Version ohne Rücksicht überschrieben wird.

Den Trick, mit `CREATE PFILE FROM SPFILE` eine temporäre herkömmliche Parameterdatei zu erzeugen, diese zu editieren und dann mit `CREATE SPFILE FROM PFILE` vor dem `STARTUP` wieder eine Serverparameterdatei zu erzeugen, benötigt man u.a. dann, wenn man einen Parameter ganz austragen will oder sich ein Fehler eingeschlichen hat, mit dem der `STARTUP` nicht mehr möglich ist – das `ALTER SYSTEM SET`-Kommando mit `SCOPE = SPFILE` funktioniert nämlich ausschließlich bei hochgefahrener Instanz.

3.8 Block-Change-Tracking-Datei

Neben den Redologs, die grundsätzlich gepflegt werden, sowie den Flashback-Logs, die optional eingeschaltet werden können, gibt es einen weiteren Mechanismus, um Änderungen der Datenbank zu protokollieren. Der Hintergrund beim so genannten Block Change Tracking ist es, inkrementelle Sicherungsstrategien zu erleichtern.

Mit dem mitgelieferten Werkzeug RMAN (Recovery Manager) kann eine Oracle-Datenbank aufbauend auf einem initialen vollständigen Backup jeweils inkrementell gesichert werden, es werden also nur die Änderungen seit dem letzten Backup gesichert (siehe Kapitel 10). Dies funktioniert insbesondere bei Datenbanken mit geringem bis mäßigen Änderungsvolumen sehr gut. Bis Oracle9i musste nichtsdestoweniger zur Erstellung des inkrementellen Backups der gesamte Datenbankinhalt gelesen werden, um alle seit dem letzten Backup geänderten Blöcke zu finden.

Mit Oracle 10g ist die Möglichkeit geschaffen worden, alle Blockänderungen von vornherein mitzuprotokollieren. Im Gegensatz zu den Redologs geht es hierbei nicht darum, die geänderten Daten selbst zu protokollieren und im Recovery-Fall nachspielen zu können, sondern es genügt – unabhängig von der Anzahl und Art der Änderungen eines Blocks –, lediglich ein Flag zu pflegen, das anzeigt, dass der Block geändert wurde. Dabei dürfen natürlich auch Einfügungen durch Direct-Loads, die in den Redolog-Dateien u.U. nicht protokolliert werden, für das Backup nicht verloren gehen. Aus diesen Anforderungen heraus erklärt sich, dass ein weiterer, eigenständiger Logging-Mechanismus, nämlich das Block Change Tracking, geschaffen wurde.

Die binäre Log-Datei enthält Flags, die für jeden 32 KB-Block der Datenbank anzeigen, ob der Block geändert wurde (die Granularität beträgt unabhängig von den tatsächlichen Oracle-Blockgrößen stets 32 KB). Daher sind der Protokollierungs-Overhead und der Umfang der anfallenden Logging-Daten minimal. Alle Log-Daten einer Datenbank werden in einer einzigen Block-Change-Tracking-Datei abgespeichert, deren Größe ca. 1/30000 der Gesamtgröße aller Datendateien beträgt (je 32 KB Daten werden auf ein Bit abgebildet, außerdem werden bis zu 8 Bitmaps zurückliegender Backups gepflegt), aufgerundet auf volle 10 MB sowie auf 320 KB pro Datendatei. Für Datenbanken bis 300 GB beträgt die Größe dieser Log-Datei also lediglich 10 MB, bis 600 GB Datenbankgröße 20 MB etc. Bei vielen, kleinen Datendateien kann sie wegen der erwähnten Aufrundung etwas größer werden. Bei mehreren Redolog-Threads (in RAC-Umgebungen) ist die so ermittelte Größe noch mit der Anzahl der aktiven Threads zu multiplizieren.

Das Block Change Tracking kann im laufenden Betrieb unter Angabe der Logging-Datei eingeschaltet werden:

```
SQL> ALTER DATABASE ENABLE BLOCK CHANGE TRACKING
     USING FILE '/disk1/verzeichnis/block_changes.chg';
```

Verwendet man Oracle Managed Files, empfiehlt es sich, die `USING`-Klausel wegzulassen. Der Zielort für die Block-Change-Tracking-Datei ergibt sich dann aus den Serverparametern `db_create_file_dest` und `db_unique_name`, als: *<db_create_ file_dest>* /*<db_unique_name>*/`changetracking`

Abgesehen von der Möglichkeit, die Log-Datei zu verschieben, ist die Verwaltung der Datei ansonsten vollständig automatisiert. Auch die Nutzung der Datei durch RMAN im Rahmen inkrementeller Sicherungen geschieht automatisch.

Über die Spalte `status` der View `v$block_change_tracking` kann abgefragt werden, ob Block Change Tracking zurzeit aktiviert ist. Die eigentliche Protokollierung wird durch den `CTWR`-Hintergrundprozess geleistet, der bei Redolog-Schreibvorgängen vom `LGWR`-Prozess angestoßen wird.

Beim inkrementellen Backup liest RMAN nur noch diejenigen Blöcke, die laut Tracking-Datei geändert wurden. Dies kann auch über die Spalte `used_change_tracking` der View `v$backup_datafile` nachvollzogen werden. Die Tracking-Datei selbst ist nicht Bestandteil der Sicherung.

Achtung: Inkrementelle Backups mit RMAN – mit oder ohne Block Change Tracking – sind nur mit der Enterprise Edition der Oracle 10g-Datenbank möglich! Außerdem sollte Block Change Tracking erst ab dem Patch-Level 10.1.0.3 verwendet werden, da die Funktionalität zuvor durch verschiedene Software-Bugs deutlich eingeschränkt war.

3.9 Physikalisches Layout

Das physikalische Layout der Datenbank entscheidet sowohl über die Performance der Datenbank als auch über Administrierbarkeit und Übersichtlichkeit. In vielen Fällen wird der Standard verwendet – ein lokales Dateisystem.

Unter MS-Windows sind NTFS, unter Unix diverse Derivate wie Reiser-FS unter Linux Beispiele hierfür. Eine grundsätzliche Empfehlung ist, immer Logging-Dateisysteme für den Betrieb von Datenbanken zu wählen, da diese gegenüber Systemabstürzen wesentlich stabiler und nach Systemabstürzen deutlich schneller wieder verfügbar sind. Also: unter MS-Windows »Finger weg von FAT«!

Es gibt zwei Gründe, kein lokales Dateisystem zu verwenden: Performance und der Einsatz von RAC.

Die Performance-Frage bezieht sich darauf, dass ein Dateisystem grundsätzlich mit einem Cache im RAM arbeitet. Dieser wird zwar von Oracle grundsätzlich nicht genutzt – zumindest alle Schreiboperationen arbeiten mit *write through cache* –, er muss jedoch trotzdem verwaltet werden und nimmt einen nicht unerheblichen Teil des RAM in Anspruch. Weicht man z.B. auf Raw Devices aus, so entfällt der Dateisystem-Cache komplett, was per se zu einem besseren IO-Durchsatz führt.

Der Dateisystem-Cache ist auch der Grund dafür, dass ein lokales Dateisystem nicht für RAC-Konfigurationen verwendet werden kann – es ist nun einmal für einen Rechner lokal und damit in diesem Moment für die anderen nicht nutzbar.[4] RAC setzt aber den gleichberechtigten, gleichartigen Zugriff für alle Rechner voraus.

3.9.1 Dateisystem

Oracle gibt im Rahmen der so genannten Optimal Flexible Architecture (OFA) bestimmte Empfehlungen hinsichtlich der Speicherung von Datendateien. Diese Empfehlungen haben sich (teilweise in leicht abgewandelter Form) in der Praxis als sehr sinnvoll herausgestellt:

- Alle Datenbankdateien außer Parameterdatei und Passwortdatei sollten auf einem anderen physikalischen Datenträger gespeichert werden als die Oracle-Software
- Für Unix: Die Mountpoints für die Datenbankdateien sind von der Form /<*mp*><*nn*>, wobei <*mp*> für einen beliebigen, kurzen String steht und <*nn*> eine laufende Nummer darstellt. Typische Mountpoints sind beispielsweise `/u01`, `/u02` … oder `/disk01`, `/disk02` …
- Für MS-Windows: Die Mountpoints für die Datenbankdateien sind von der Form *X*`:\oracle`, wobei *X* für einen Laufwerksbuchstaben steht. Typische Mountpoints sind beispielsweise `D:\oracle`, `E:\oracle` etc.

4 Die Verwendung von NFS-Filesystemen ist nur in Ausnahmefällen mit spezieller Hardware erlaubt (z.B. Network Applicance oder kurz NetApp).

- Unterhalb eines Mountpoints befindet sich ein Verzeichnis `oradata` und darunter ein Verzeichnis mit dem Datenbanknamen (Serverparameter `db_name` bzw. `db_unique_name` für Oracle 10g-Datenbanken). Dies ist das Verzeichnis für die Datenbankdateien, beispielsweise `/u01/oradata/PDWH10G` für eine Datenbank mit dem Namen `PDWH10G`.
- Redolog- und Archivelog-Dateien sollten sich auf einem anderen physikalischen Datenträger als die Datendateien befinden, damit im Fall eines Medienfehlers des Datenträgers mit den Datendateien eine Wiederherstellung mit Hilfe der Log-Dateien möglich ist.
- Wird die Kontrolldatei gespiegelt, sollten sich die verschiedenen Kontrolldateien auf unterschiedlichen physikalischen Datenträgern befinden, damit im Fall eines Medienfehlers nicht alle Kontrolldateien beschädigt sind.
- Zwecks besserer Unterscheidbarkeit sollten für Datenbankdateien die folgenden Dateiendungen verwendet werden:
  - `.dbf` für Datendateien (auch für System-, SysAux- und Undo-Tablespaces),
  - `.tmp` für Temporärdateien (Temporär-Tablespaces)
  - `.log` für Online-Redolog-Dateien
  - `.arc` für archivierte Redolog-Dateien (einstellbar über den Serverparameter `log_archive_format`)
  - `.ctl` für Kontrolldateien (einstellbar über den Serverparameter `control_files`)
  - `.bct` für Block-Change-Tracing-Dateien (einstellbar bei der Aktivierung des Block Change Trackings)
  - `.flb` für Flashback-Logs (werden automatisch erstellt und verwaltet)
- Daten- und Temporärdateien sollten im Dateinamen den Namen des Tablespaces sowie eine zweistellige fortlaufende Nummer enthalten, z.B. `system01.dbf`, `temp01.tmp` etc.
- Online-Redolog-Dateien sollten das Wort »`redo`« sowie die Gruppen- und Member-Nummer enthalten, also beispielsweise `redo_02_1.log` für den ersten Member der zweiten Redolog-Gruppe. Werden ausschließlich Hardware- oder RAID-Spiegelungen benutzt, kann die Member-Nummer auch weggelassen werden.
- Archivierte Redolog-Dateien sollten den Datenbanknamen (Serverparameter `db_name` bzw. `db_unique_name` für Oracle 10g-Datenbanken), die so genannte Log-Sequence-Number sowie die Nummer des Redolog-Threads enthalten. Dies wird über die Einstellung des Serverparameters `log_archive_format` erreicht. Für Oracle 10g-Datenbanken sollte außerdem unbedingt die Resetlogs-ID mit angegeben werden, um ein Recovery zu vereinfachen. Empfehlenswert ist hier beispielsweise für eine Datenbank mit dem Namen `PDWH10G` der Wert: `log_archive_format = PDWH10G_%r_%t_%s.arc`.

Oracle Managed Files

Seit Oracle9i kann die Verwaltung von Datenbankdateien auf Wunsch auch weitestgehend der Datenbank überlassen werden. Zu diesem Zweck müssen Standardpfade für neue Datenbankdateien konfiguriert werden. Anschließend können bei der Erstellung von Tablespaces, beim Hinzufügen von Daten-, Temporär- oder Redolog-Dateien oder auch bei der Erstellung einer neuen Datenbank die Klauseln, die sich auf Dateinamen und Größen beziehen, einfach weggelassen werden. Zum Beispiel:

```
SQL> CREATE TABLESPACE users;
SQL> ALTER TABLESPACE users ADD DATAFILE;
SQL> ALTER DATABASE ADD LOGFILE GROUP 4;
SQL> CREATE DATABASE;
```

Listing 3.11: Listing 3.1: Beispiele für Oracle Managed Files

Für die verschiedenen Arten von Datenbankdateien können unterschiedliche Pfade konfiguriert werden. Dabei spielen folgende Serverparameter eine Rolle:

- Der Parameter `db_create_file_dest` gibt ein Basisverzeichnis für Daten- und Temporärdateien an.
- Die Parameter `db_create_online_log_dest_`<*n*> (<*n*> = 1, 2, 3, 4, 5) geben bis zu fünf Basisverzeichnisse für Redolog-Gruppen und Kontrolldateien an. Die angegebenen Pfade sollten auf jeweils unterschiedlichen physikalischen Datenträgern und auf einem anderen Datenträger als `db_create_file_dest` liegen. Hintergrund ist, dass die einzelnen Member (Spiegel) einer Redolog-Gruppe sowie die Spiegel der Kontrolldatei entsprechend auf die angegebenen Pfade verteilt werden.
- Der Parameter `db_recovery_file_dest` ist erst ab Oracle 10g definiert und gibt ein Basisverzeichnis für Flashback-Logs und archivierte Redolog-Dateien an.
- In Oracle9i werden die entsprechenden Dateien als Oracle Managed Files direkt in den beschriebenen Basisverzeichnissen abgespeichert. In Oracle 10g hat man etwas mehr Ordnung hereingebracht. In dem Basisverzeichnis wird zunächst ein Unterverzeichnis mit dem Datenbanknamen angelegt (Serverparameter `db_unique_name`), hierin wird ein weiteres Unterverzeichnis gemäß dem Dateityp (z.B. `onlinelog` für die Online-Redolog-Dateien) angelegt. Der letzte Schritt entspricht zwar formal nicht den OFA-Regeln, bringt aber jedenfalls noch etwas mehr Übersichtlichkeit.

Abbildung 3.6 zeigt noch einmal zusammenfassend die Speicherung von Datenbankdateien als Oracle Managed Files in Oracle 10g. Der grau unterlegte Bereich gilt für Oracle9i.

Dateityp	Datei-endung	Basisverzeichnis nach Erstellung			Unterverzeichnis (für Oracle 10g)
		db_create_online_log_dest_n	db_create_file_dest	db_recovery_file_dest	
Datendateien / Temp-Dateien	.dbf .tmp		X		<db_unique_name>/datafile/
Online Redologs	.log	X	(X)	(X)	<db_unique_name>/onlinelog/
Kontrolldateien	.ctl	X	(X)	(X)	<db_unique_name>/controlfile/
Archivierte Redologs	.arc			X	<db_unique_name>/archivelog/<yyyy_mm_dd>/
Block-Change-Tracking-Dateien	.bct		X		<db_unique_name>/changetracking/
Flashback Logs	.flb			X	<db_unique_name>/flashback/
RMAN Backup-Sets	.bkp			X	<db_unique_name>/backupset/
RMAN Image Copies	.dbf ...				<db_unique_name>/datafile/ ...

Abbildung 3.6: Speicherorte und Dateiendungen für Oracle –Managed Files

Neben den beschriebenen Standardpfaden ist auch die Namensgebung für Oracle – Managed Files festgelegt. Diese Dateien haben stets Namen der Form `o1_mf_<id>_.<endung>`. Dabei steht *<endung>* für eine der aufgeführten Dateiendungen, *<id>* ist eine Zeichenkette, die einerseits eine eindeutige Beziehung zu dem zugrunde liegenden Objekt schaffen, andererseits den Dateinamen auch eindeutig machen soll.

Für Datendateien ist *<id>* beispielsweise von der Form *<tsname>_<uid>*, wobei *<tsname>* der auf acht Zeichen abgeschnittene Name des Tablespaces und *<uid>* eine eindeutige achtstellige Zeichenkombination darstellt. z.B.:

```
SQL> CREATE TABLESPACE daten_big;
SQL> SELECT file_name
     FROM   dba_data_files
     WHERE  tablespace_name = 'DATEN_BIG';

FILE_NAME
------------------------------------------------------------
D:\ORACLE\ORADATA\PS10\DATAFILE\O1_MF_DATEN_BI_1GBKDX9R_.DBF
```

Nach der Erstellung können OMF wie andere Dateien auch behandelt werden, insbesondere können sie umbenannt und verschoben werden. Für den Status als OMF ist nicht der Pfad, sondern ausschließlich der Dateiname relevant. Jede Datei mit einem Dateinamen der Form `o1_mf_*_.<endung>` wird von Oracle als OMF identifiziert. Das explizite Anlegen einer solchen Datei z.B. als Datendatei oder Redolog-Datei ist nicht erlaubt und wird mit einer Fehlermeldung quittiert, z.B.:

```
SQL> CREATE TABLESPACE daten_big
     DATAFILE
     'D:\ORACLE\ORADATA\PS10\O1_MF_DATEN_BI_1GBKDX9R_.DBF';
CREATE TABLESPACE daten_big
*
FEHLER in Zeile 1:
ORA-01276: Datei
D:\ORACLE\ORADATA\PS10\O1_MF_DATEN_BI_1GBKDX9R_.DBF kann nicht
hinzugefügt werden. Datei hat Name von Oracle Managed Files.
```

Ein nachträgliches Umbenennen einer Datendatei in eine OMF-Datei und auch ein nachträgliches Verschieben einer OMF-Datei in ein anderes Verzeichnis ist jedoch möglich, z.B.:

```
SQL> ALTER TABLESPACE daten_big OFFLINE;
Tablespace wurde geändert.
SQL> -- Verschieben der Datendatei auf Betriebssystemebene
SQL> ALTER DATABASE RENAME FILE
   'D:\ORACLE\ORADATA\PS10\DATAFILE\O1_MF_DATEN_BI_1GBLCKON_.DBF'
     TO
     'D:\ORACLE\ORADATA\PS10\O1_MF_DATEN_BI_1GBLCKON_.DBF';
```

Letztlich ist jedoch die entscheidende Frage, worin die Vorteile von Oracle Managed Files besteht und wie man diese Funktionalität am besten einsetzen sollte.

- OMF vereinfachen die Erstellung und Erweiterung von Tablespaces und Redolog-Gruppen. Setzt man dies konsequent ein, werden sämtliche Datenbankdateien außer Parameter- und Passwortdatei automatisch erzeugt und benannt. Dies ist vor allem für Test- und Entwicklungsumgebungen ein Vorteil. In produktiven Umgebungen ist die automatische Erstellung von Daten- und Temporärdateien nur sinnvoll, wenn es einen einzigen Mountpoint gibt, der über die darunter liegende Hardware eine I/O-Parallelisierung durchführt. Liegen dagegen mehrere Mountpoints vor bzw. wird die I/O-Verteilung eher manuell durch geschicktes Verteilen der Tablespaces über die Mountpoints durchgeführt, kommen OMF dafür nicht in Frage.
- Das versehentliche Erstellen einer Datendatei in einem nicht dafür vorgesehenen Verzeichnis ist ausgeschlossen.
- Beim Löschen eines Tablespaces oder einer Redolog-Gruppe werden die zugehörigen Dateien auf Betriebssystemebene automatisch gelöscht. Das versehentliche Löschen einer falschen Datei ist dadurch ausgeschlossen. Für Daten- und Temporärdateien gibt es diese Möglichkeit allerdings bereits durch die Klausel `INCLUDING CONTENTS AND DATAFILES` des `DROP TABLESPACE`-Kommandos. Außerdem kommt das Löschen eines Tablespaces im produktiven Betrieb nicht häufig vor, wohl aber in Test- oder Entwicklungsumgebungen.

- SQL-Skripte, die Tablespace-Definitionen enthalten, können plattformunabhängig geschrieben werden. Lediglich die genannten Serverparameter müssen zuvor plattformabhängig gesetzt werden.
- Sie verhindern häufig fruchtlose Diskussionen über geeignete Namensgebungen für Datenbankdateien.

Zusammenfassend lässt sich sagen, dass OMF für Test- und Entwicklungsumgebungen eine gute Wahl darstellen sowie für Produktionsumgebungen, die über ein leistungsfähiges SAN-Storage verfügen, das dem DBA Fragen bzgl. I/O-Verteilung abnimmt. Ggf. sollte trotzdem eine `SIZE`-Klausel mit angegeben werden (Standardgröße für Daten-, Temporär- und Redolog-Dateien ist 100MB), auf jeden Fall sollte im Rahmen der `AUTOEXTEND`-Klausel eine Maximalgröße gesetzt werden, da OMF sonst mit unbegrenztem Wachstum erstellt werden. Für Oracle9i muss unbedingt auch die Erweiterungsgröße (`NEXT`-Klausel) vom Standardwert (ein Block bei Oracle9i, 100 MB bei Oracle 10g) auf einen sinnvollen Wert geändert werden, also z.B.:

```
SQL> CREATE TABLESPACE users
     DATAFILE SIZE 1000M AUTOEXTEND ON NEXT 100M MAXSIZE 5000M;
SQL> ALTER TABLESPACE users ADD
     DATAFILE SIZE 1000M AUTOEXTEND ON NEXT 100M MAXSIZE 5000M;
```

3.9.2 Cluster-Dateisystem

Cluster-Dateisysteme werden meist vom Systemhersteller als Zusatzoption für Cluster-Systeme angeboten; für Linux existiert das Projekt Oracle Cluster File System im Rahmen des Oracle Technology Networks, das unter GNU General Public License (GPL) veröffentlicht wird.

Cluster-Dateisysteme werden wie lokale Dateisysteme in den Verzeichnisbaum integriert, mit dem Unterschied, dass dies auf mehreren Rechnern gleichzeitig passieren kann. Damit können sie für RAC-Konfigurationen verwendet werden. Dadurch dass bei Cluster-Dateisystemen im Normalfall wiederum Dateisystem-Caches vorhanden sind, sind sie in den meisten Fällen langsamer als Raw Devices; allerdings sind die Vorzüge bei der Administrierbarkeit nicht zu unterschätzen.

Bei einigen Cluster-Dateisystemen ist es möglich, die Installation der Oracle Software im Cluster-Dateisystem durchzuführen. Damit wird der Installationsaufwand minimiert, z.B. auch für neue Knoten, die ins Cluster integriert werden sollen, allerdings zum Preis einer weniger flexiblen Konfiguration. Für Umgebungen mit Forderungen für absolut kleine Auszeiten bei Wartungsarbeiten ist eine lokale Installation auf jedem einzelnen Knoten vorzuziehen. Dieses Konzept wird vom Univeral Installer ebenfalls komfortabel unterstützt.

Ansonsten ist über Cluster-Dateisysteme wenig Allgemeines zu sagen – dazu sind die einzelnen Lösungen zu systemspezifisch.

3.9.3 Raw Devices

Raw Devices sind eine Spezialität von Unix-Systemen und seit langer Zeit die Alternative zu Dateisystemen. Übrigens bieten viele Systeme mittlerweile einen Logical Volume-Manager an, der wesentlich mehr Komfort für Raw Devices bietet als das reine Betriebssystem. Mit einem Logical Volume-Manager lassen sich Logical Volumes erzeugen, für die praktisch alle Aussagen über Raw Devices ebenfalls zutreffen.

Bei einem normalen Filesystem, wie es unter Unix und Linux in diversen Versionen zur Verfügung steht, wird ein physikalischer Plattenbereich, ein so genanntes Device, entsprechend formatiert, z.B. mit dem Befehl `newfs`. Dieses Filesystem wird dann als Mountpoint angeschlossen und kann jetzt von jedem, der die Berechtigung dazu hat, mit Dateien gefüllt werden. Ein Buffercache im Memory sorgt dafür, dass die wichtigsten Daten sowie die Dateisystemstruktur immer im Hauptspeicher bleiben und somit eine optimale Performance gewährleistet ist. Wenn man sich die Oracle-Datenbankstruktur ansieht, stellt man fest, dass Oracle genauso funktioniert: Mit dem Befehl `CREATE TABLESPACE … DATAFILE …` wird ein entsprechender Bereich formatiert (in Oracle-Blöcken), und über die SGA wird ein Buffercache im Memory zur Verfügung gestellt. Also stellt sich die Frage: Wofür braucht man noch ein Filesystem? Ganz im Gegenteil, bei Schreiboperationen muss verhindert werden, dass die Daten im Betriebssystemcache hängen bleiben. Somit ist dieser für Oracle fast wertlos. Also ist es unter Umständen sinnvoller, die Oracle-Datendateien direkt auf die Raw Devices zu platzieren und damit den Overhead des Betriebssystems zu eliminieren. Das wird jedoch mit einer Einschränkung in der Flexibilität erkauft. Raw Devices haben eine bestimmte Größe, die, wenn überhaupt, nur mit Betriebssystemkommandos geändert werden kann. Je Raw Device kann außerdem nur genau eine Oracle-Datei benutzt werden, so dass von vornherein klar sein muss, wie groß die Dateien werden. Ein `AUTOEXTEND` der Datendatei ist also nicht möglich. Außerdem ist dem Betriebssystem oft nicht bekannt, dass die Dateien in Benutzung sind, somit kann jederzeit mit dem Befehl `newfs` ein Filesystem auf das Raw Device angelegt und damit die Datenbank zerstört werden. An dieser Stelle empfehlen sich Logical Volumes, deren Namen man bei der Erstellung bestimmen kann. Hier sollte man für Oracle Logical Volumes Standards wie `lora_<dbname>_xxx` benutzen.

Ein weiterer gerne genutzter Trick ist, die Raw Devices nicht direkt als Datenbankdateien zu verwenden, sondern ein Verzeichnis zu erstellen, das symbolische Links auf die Raw Devices enthält. Die Namen der symbolischen Links können dann wieder sprechend bezüglich ihrer Bedeutung in der Datenbank sein. Die symbolischen Links (inklusive des vollständigen Pfads) werden dann als Namen für die Datenbankdateien verwendet. Mit einem Listing des Verzeichnisses hat der Administrator nun eine einfache Möglichkeit, die verwendeten Raw Devices abzufragen.

Obwohl Raw Devices grundsätzlich schnellere IOs erlauben als Dateisysteme, sollten sie nicht ohne Vorüberlegungen eingesetzt werden. Die Administration einer Oracle-Datenbank mit Raw Devices ist um einiges komplexer als bei der Verwendung von normalen Dateien. Weiterhin werden die Vorteile im IO-Bereich oft gar nicht bemerkt, da z.B. alle DBW0-IOs asynchron zu den SQL-Befehlen im Hintergrund laufen. Solange das IO-System nicht überlastet ist, sind bei den Anwendungen keine verbesserten Antwortzeiten messbar. Den größten messbaren Vorteil

erzielt man oft bei der Verwendung von Raw Devices für Redolog-Member bei Systemen mit hoher Transaktionslast, da `COMMIT`-Befehle synchron mit IOs des LGWR-Prozesses laufen.

3.9.4 Automatic Storage Management (ASM)

Mit Oracle 10g führt Oracle ein eigenes Filesystem (ASM, Automatic Storage Management) ein, das unabhängig vom Betriebssystem (z.Z. Unix und MS-Windows) die optimale Implementierung für den Betrieb von Oracle-Datenbanken darstellen soll. Neben der Möglichkeit der Mehrfachspiegelung werden die Filesysteme über alle verfügbaren Devices gestriped und entsprechen damit der von Oracle seit längerem vertretenen These S.A.M.E (Stripe and Mirror Everything).

Ein paar Worte zur Historie: Oracle hat sich im Umfeld Oracle Parallel Server und Real Application Clusters in der Vergangenheit schwer getan, die Storages zu verwalten. Zunächst wurden nur Raw Devices unterstützt, was dazu führte, dass nur mit fest vorgegebenen Dateigrößen gearbeitet werden konnte. Mit Version 8i wurde dann auf einigen Plattformen (z.B. AIX) das von den Betriebssytemherstellern entwickelte Global File System unterstützt. Leider war hier das Problem »auf einigen Plattformen«. Mit Version 9i hat Oracle dann das Oracle Cluster File System (OCFS), also eine eigene Implementierung des Global File Systems eingeführt. Dieses ist betriebssystemunabhängig und kann als separates Produkt eingesetzt werden, die Verzeichnisse sind aber z.B. unter MS-Windows als normale Dateisysteme zu erkennen, und somit besteht die Gefahr – wie wir sie schon erlebt haben – , dass diese Verzeichnisse durch Unachtsamkeit zerstört und damit die Datenbank korrumpiert wird. Außerdem steht OCFS nur für den Betrieb von Real Application Clusters zur Verfügung. Mit Version 10g wird jetzt mit ASM ein unabhängiges Produkt geschaffen, das für alle Oracle-Datenbanken zur Verfügung steht und für das Betriebssystem unsichtbar ist.

Funktionalität

ASM besteht aus einer eigenen Instanz, in der die Disk-Devices verwaltet werden. Eine Oracle-Datenbank nimmt über den ebenfalls in Oracle 10g neu eingeführten Oracle Cluster Synchronization Service (OCSS) eine Verbindung zur ASM-Instanz auf und erhält über diesen die entsprechenden Ressourcen, die vorher eingerichtet worden sind. Intern werden diese Ressourcen wie bei einem normalen Filesystem in Verzeichnissen verwaltet. Dabei werden die Devices speziell formatiert und stehen somit dem Betriebssystem nicht mehr zur Verfügung. Die Gefahr einer unbedachten Formatierung ist dadurch minimiert. Da es sich hierbei um eine interne Kommunikation handelt, muss unter MS-Windows der Parameter `SQLNET.AUTHENTICATION_SERVICES= (NTS)` in der Datei `sqlnet.ora` gesetzt sein.

Das Zusammenspiel von Datenbank- und ASM-Instanz über den OCSS-Daemon hat den Nachteil, wenn die ASM-Instanz oder der OCSSD Prozess abstürzt, werden die beteiligten Datenbanken ebenfalls beendet.

Die ASM-Instanz wird standardmäßig über die `ORACLE_SID` »`+ASM`« angegeben. Folgende Parameter sind dabei notwendig:

```
instance_type       ='ASM'
asm_diskgroups      ='ORA_ASM'
```

Nähere Informationen zu den Instanzparametern erhalten Sie im Kapitel 4.4.

Aufbauen einer Oracle-Datenbank mit ASM

Um eine ASM-Instanz aufzubauen, werden unformatierte Devices benötigt. Wenn ein Unix- bzw. Linux-Betriebssystem verwendet wird, reicht es, wenn die Devices dem Oracle-Benutzer und der Gruppe gehören. Außerdem darf hier der erste Block der physikalischen Platte nicht genutzt werden. Zum Löschen der ASM-Informationen müssen die ersten Blöcke jeder Partition überschrieben werden, dies geschieht z.B. mit dem Befehl:

```
dd if=/dev/zero of=/dev/rdsk/c3t0d0s4 bs=8192 count=100
```

Listing 3.12: Löschen der ASM-Informationen unter Unix

Unter Windows müssen die Partitionen vorhanden, dürfen jedoch nicht mit Betriebssystemmitteln formatiert worden sein. Bevor sie als Diskgruppen benutzt werden können, müssen sie mit dem Werkzeug *asmtool* (Kommandozeile) oder *asmtoolg* (grafische Oberfläche) vorformatiert (stamp) werden. Diese Partitionen stehen jetzt für das Betriebssystem nicht mehr zur Verfügung. Um sie wieder »normal« nutzbar zu machen, müssen Sie wiederum das ASM-Tool aufrufen und die Stempel löschen. Das folgende Bild zeigt das Tools *asmtoolg* mit der Anzeige der möglichen Partitionen und deren Größe.

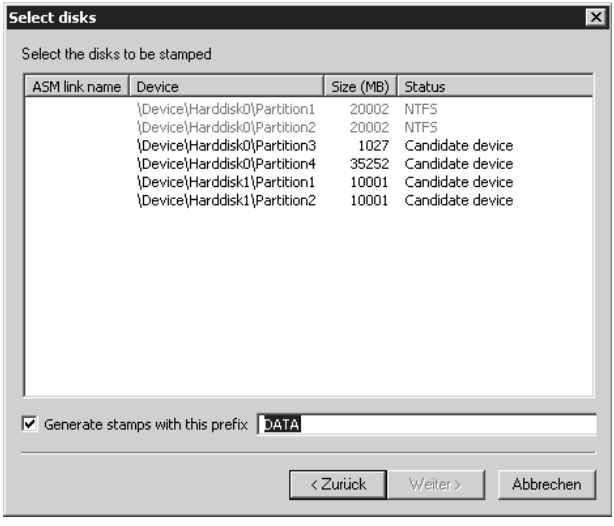

Abbildung 3.7: asmtoolg

Sobald die Partitionen für ASM genutzt werden, haben sie zusätzlich einen *ASM Link Name*, der sich aus »ORCLDISK«, dem Präfix und einer eindeutigen Nummer zusammensetzt.

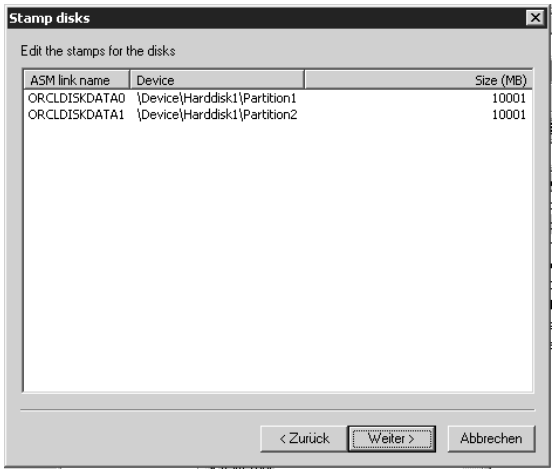

Abbildung 3.8: ASM Link Name

Bei der Verwendung des Database Configuration Assistants (DBCA) für die Erstellung einer Datenbank können Sie *Automatic Storage Management* als Speicheroption auswählen.

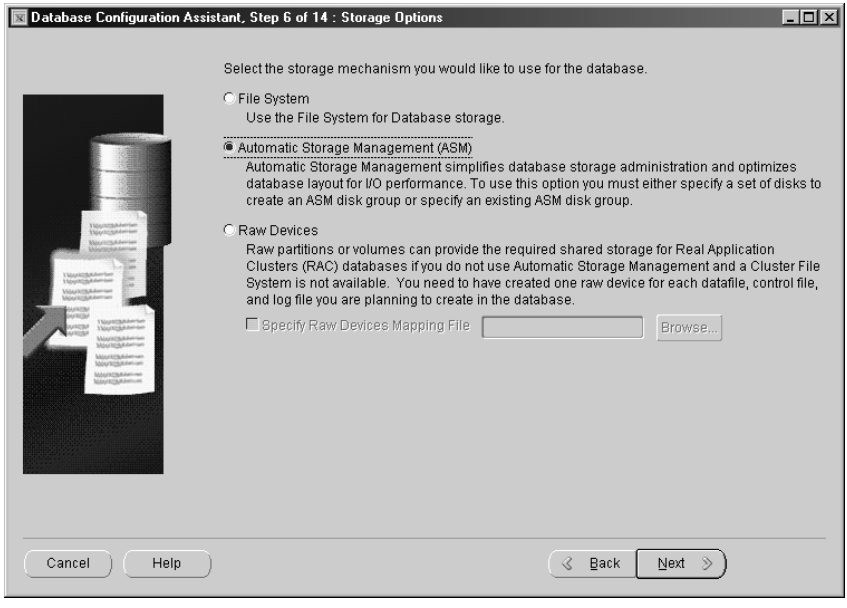

Abbildung 3.9: ASM-Installation über DBCA

Sollte es hier noch keine entsprechende ASM-Instanz geben, wird diese jetzt automatisch angelegt und hochgefahren, dafür können Sie in einem weiteren Schritt das SYS-Passwort und die Serverparameter angeben. Als Nächstes können dann die Diskgruppen angelegt bzw. verwaltet werden.

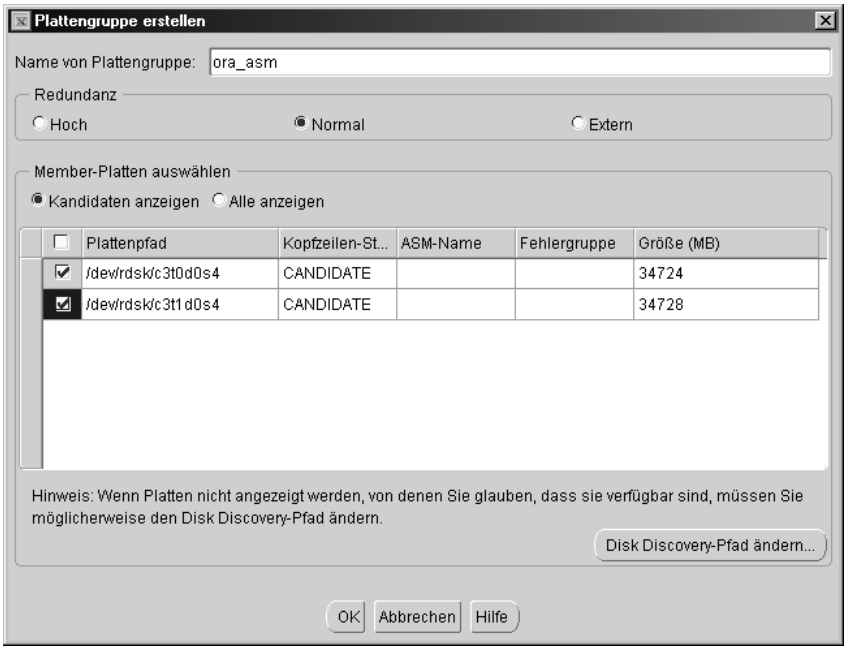

Abbildung 3.10: Erstellen einer Diskgruppe

Folgende Parameter müssen angegeben werden:

- Name der Diskgruppe (hier ora_asm)
- Redundanz, hier sind folgende Optionen möglich:
  - Hoch: dreifache Spiegelung
  - Normal: einfache Spiegelung
  - Extern: keine Spiegelung
- Die möglichen Devices (Kandidaten). Dabei sollt ein Vielfaches der gewählten Redundanz an Devices zur Verfügung stehen, da ASM sonnst auf der gleichen Platte spiegelt (es ist theoretisch möglich, eine dreifache Spiegelung mit nur einer Diskgruppe durchzuführen).

Eine Größe für Striping kann an dieser Stelle nicht angegeben werden, es gibt für die Verwendung von ASM nur zwei Größen: *FINE* mit 128 KB und *COARSE* mit 1 MB.

Wenn an dieser Stelle die Devices nicht angezeigt werden, kann das unterschiedliche Gründe haben, einige davon sind:

- Unter MS-Windows wurden die Devices nicht mit dem `asmtool` vorformatiert.
- Unter Unix stimmen die Berechtigungen (Oracle-Benutzer und -Gruppe) nicht.
- Der Pfad, unter dem die Devices gesucht werden sollen (Serverparameter `asm_disk_string`), ist gesetzt, zeigt aber auf einen anderen Bereich).

Sind alle Eingaben korrekt, wird als Nächstes die Diskgruppe angelegt, und im DBCA werden die verfügbaren Diskgruppen angezeigt.

Abbildung 3.11: Konfigurierte Diskgruppe

Unschönerweise sind die Größen (*Größe* und *Frei*) nicht entsprechend der Redundanz, sondern als Summe der verfügbaren Devices angegeben; in diesem Fall also ca. 70 GB anstelle der tatsächlich zur Verfügung stehenden 35 GB.

Als Speicherort für die Datendateien werden jetzt Oracle Managed Files mit dem Namen der ASM-Instanz angeboten (`+ORA_ASM`). Die weitere Vorgehensweise für die Konfiguration der Datenbank entspricht der für Oracle Managed Files.

ASM-Layout

Bei einer Standardinstallation werden alle Komponenten der Datenbank im ASM-Filesystem abgelegt. Dazu gehören:

- Datendateien
- Kontrolldateien
- Online-Redolog-Dateien
- Temporäre Dateien
- Parameterdateien
- Flashback-Logs
- Backup-Sets
- Archivierte Redolog-Dateien

Diese werden in den zugehörigen Unterverzeichnissen, die automatisch angelegt werden, abgespeichert. Die gesamte Speicherung erfolgt als Oracle Managed Files (OMF), kann aber zusätzlich mit eigenen Namen und Verzeichnissen erweitert werden. Wenn keine speziellen Einstellungen über Templates vorgenommen werden, sind die Kontroll- und Online-Redolog-Dateien mit 128 KB Stripegröße (*Fine*) eingestellt, alle anderen Bereiche mit 1 MB (*Coarse*).

Eine fertige mit ASM-verwaltete Datenbank sieht also z.B. wie folgt aus:

```
SQL> SELECT tablespace_name, file_name, bytes/1024/1024 MByte
     FROM   dba_data_files;
TABLESPACE_NAME  FILE_NAME                                    MB
---------------- ------------------------------------- -------
SYSTEM           +ORAASM1/sunasm/datafile/system.264.1      300
UNDOTBS1         +ORAASM1/sunasm/datafile/undotbs1.265.1    500
SYSAUX           +ORAASM1/sunasm/datafile/sysaux.266.1      230
USERS            +ORAASM1/sunasm/datafile/users.268.1         5
DEMO             +ORAASM1/sunasm/datafile/demo.270.1       1000
SQL> SELECT name
     FROM   v$controlfile;
NAME
-----------------------------------------
+ORAASM1/sunasm/controlfile/current.256.1
+ORAASM1/sunasm/controlfile/current.257.1
```

Listing 3.13: Mit ASM verwaltete Datenbank

Diese Information kommt wie üblich aus der Datenbank (in diesem Beispiel heißt diese `SUNASM`. Um die gleichen Daten auch in der ASM-Instanz darstellen zu können, bedarf es eines etwas umfangreicheren SQL-Befehls:

```
SELECT
'+'||dg.name||'/'||sid.name||'/'||typ.name||'/'||datei.name Filename,
       round(info.bytes/1024/1024) MByte
FROM   v$asm_diskgroup dg,
       v$asm_alias sid,
       v$asm_alias typ,
       v$asm_alias datei,
       v$asm_file  info
WHERE  sid.group_number    = dg.group_number
AND    typ.group_number    = dg.group_number
AND    datei.group_number  = dg.group_number
AND    sid.reference_index = typ.parent_index
AND    typ.reference_index = datei.parent_index
AND    datei.file_number   = info.file_number;
```

```
FILENAME                                                    MBYTE
---------------------------------------------------------- -----
+ORA_ASM1/SUNASM/CONTROLFILE/Current.256.565036357              2
+ORA_ASM1/SUNASM/ONLINELOG/group_1.258.565036359               20
+ORA_ASM1/SUNASM/ONLINELOG/group_2.259.565036359               20
+ORA_ASM1/SUNASM/ONLINELOG/group_3.260.565036361               20
+ORA_ASM1/SUNASM/ONLINELOG/group_4.261.565036363               20
+ORA_ASM1/SUNASM/DATAFILE/SYSTEM.262.565036365               210
+ORA_ASM1/SUNASM/DATAFILE/INDX.268.565036827                 100
+ORA_ASM1/SUNASM/DATAFILE/UTILS.267.565036827                 10
+ORA_ASM1/SUNASM/DATAFILE/TSUNDO.263.565036381               100
+ORA_ASM1/SUNASM/DATAFILE/SYSAUX.264.565036385               200
+ORA_ASM1/SUNASM/DATAFILE/DATA.266.565036397                 100
+ORA_ASM1/SUNASM/TEMPFILE/TEMP.265.565036397                  50
+ORA_ASM/SUNASM/PARAMETERFILE/spfile.269.565036399             0
+ORA_ASM/SUNASM/FLASHBACK/log_1.270.565036406                  8
+ORA_ASM/ARCHIVELOG/2004_06_23/thread_1_seq_50.271.565036415   9
+ORA_ASM/ARCHIVELOG/2004_06_23/thread_1_seq_51.272.565036416   0
```

Listing 3.14: Datenbankdateien aus ASM-Sicht

Manuelle ASM-Konfiguration

Natürlich kann eine ASM-Instanz auch manuell aufgesetzt werden. Dafür muss zunächst, ähnlich wie beim Anlegen einer Datenbank, eine leere Instanz hochgefahren und anschließend müssen die Diskgruppen erzeugt werden.

Als Beispiel wird folgende Initialisierungsdatei mit dem Namen `init+ASM.ora` im `$ORACLE_HOME/dbs` Verzeichnis erstellt.

```
*.instance_type              ='ASM'
*.asm_diskgroups             ='ORA_ASM'
*.background_dump_dest       ='/oracle/admin/+ASM/bdump'
*.core_dump_dest             ='/oracle/admin/+ASM/cdump'
*.large_pool_size            =12M
*.remote_login_passwordfile  ='EXCLUSIVE'
*.user_dump_dest             = '/oracle/admin/+ASM/udump'
```

Listing 3.15: Initialisierungsdatei der ASM-Instanz

Als Nächstes wird die Datenbank gestartet:

```
% set ORACLE_SID=+ASM
% sqlplus / as sysdba
SQL> startup
```

Listing 3.16: Starten der ASM-Instanz

Jetzt können die Diskgruppen mit dem Befehl `CREATE DISKGROUP` angelegt werden.

```
SQL> CREATE DISKGROUP ora_asm NORMAL
     DISK '\\.\ORCLDISKDATA1',
          '\\.\ORCLDISKDATA2';
```

Listing 3.17: Anlegen einer Diskgruppe

Neben dem Befehl CREATE DISKGROUP gibt es nur noch die Befehle ALTER DISKGROUP und DROP DISKGROUP; insofern eine überschaubare Anzahl von Befehlen für die Verwaltung von ASM-Instanzen. Dennoch ist die Verwendung des DBCA der manuellen Erstellung vorzuziehen, da dort alle verfügbaren Devices mit der entsprechenden Größe automatisch angezeigt werden.

Verwalten von Tablespaces bzw. Datendateien über ASM

Die Verwaltung von Tablespaces über ASM entspricht der von Oracle Managed Files. Das bedeutet, Sie geben in der Regel keinen Dateinamen an, wenn Sie den Tablespace anlegen. Beim Löschen desselben werden dann die zugehörigen Datendateien automatisch gelöscht.

```
SQL> CREATE TABLESPACE blub
     DATAFILE SIZE 100M;
```

Listing 3.18: Anlegen eines Tablespaces

```
SQL>DROP TABLESPACE blub;
```

Listing 3.19: Löschen des Tablespaces

Wenn Sie jedoch explizit einen Namen für die Datendatei angeben, werden in der ASM-Instanz zwei Einträge generiert, der eine mit der ASM-eigenen Namensgebung und ein zweiter Eintrag (wie ein Link) mit Ihrem gewählten Namen. Beim Löschen des Tablespaces müssen Sie nachfolgend beide Einträge für die Datendateien mit dem Befehl ALTER DISKGROUP separat löschen. Das folgende Beispiel verdeutlicht dies:

```
SQL> CREATE TABLESPACE blub
     DATAFILE '+ORA_ASM/SUNASM/DATAFILE/blub01.dbf'
     SIZE 100M;
SQL> DROP TABLESPACE blub;
SQL> ALTER DISKGROUP ora_asm
     DROP FILE '+ORA_ASM/SUNASM/DATAFILE/blub01.dbf';
SQL> ALTER DISKGROUP ora_asm
     DROP FILE '+ORA_ASM/SUNASM/DATAFILE/blub01.dbf'
SQL> ALTER DISKGROUP ora_asm
     DROP FILE '+ORA_ASM/SUNASM/DATAFILE/BLUB.288.565038087'
```

Listing 3.20: Manuelles Anlegen und Löschen eines Tablespaces

Über den Bereich ADMINISTRATION und SPEICHERUNG kann im Enterprise Manager der Menüpunkt PLATTENGRUPPEN ausgewählt werden, mit dem man sich an einer ASM-Instanz anmelden kann. Hier kann man sich dann z.B. die Konfiguration und Platzbelegung sowie Performance-Informationen ansehen.

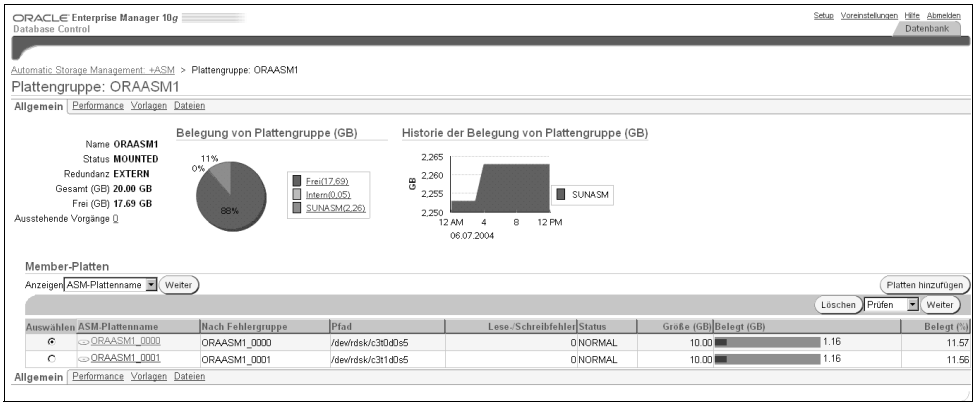

Abbildung 3.12: ASM-Darstellung im Oracle Enterprise Manager

Allerdings sollten Sie auch hier bei der Angabe der Größen bedenken, dass die Redundanz nicht berücksichtigt ist, der maximal verfügbare Platz also im obigen Fall nicht 20 GB, sondern nur 10 GB beträgt.

3.10 Ältere Funktionen

3.10.1 Dictionary-Managed Tablespaces

In den Datenbankversionen bis einschließlich Oracle 8.0 wurden alle Informationen zu Datenbankobjekten – insbesondere auch für die Verwaltung der Tablespaces – im Data Dictionary gespeichert. Das bedeutet, dass eine Erweiterung von Tabellen oder Indizes um neue Extents oder die Freigabe von Extents mit dem Befehl `TRUNCATE` zusätzliche rekursive Operationen im Kontext des Data Dictionarys durchführen. Die entsprechenden Änderungen in der Extent-Verwaltung erzeugen damit zusätzliche Rollback- und Redolog-Information.

Aus diesem Grund kann die Data-Dictionary-Verwaltung bei umfangreichen Operationen wie zum Beispiel dem Laden großer Tabellen oder der Erstellung von Indizes zu einem Performance-Engpass führen. Mit Oracle Version 7.3 konnte der Temporär-Tablespace aus dieser Verwaltung herausgetrennt werden, wodurch sich Sortierungen und auch die generelle Verwaltung beschleunigen ließen. Mit der Version 8.1 wurde dieser Schritt durch die Einführung von Locally-Managed-Tablespaces konsequent weitergeführt, um mit der Version 9 dann die Möglichkeit zu schaffen, Tablespaces auf unterschiedliche Anforderungen (freie Definition der Oracle-Blockgröße pro Tablespace) anzupassen.

Die Tablespaces des alten Typs sollten spätestens mit der Version Oracle 10g nicht mehr benutzt werden. Eine Migration bestehender DM-Tablespaces erfordert zunächst eine Abschätzung darüber, ob die Migration online, d.h. im laufenden Betrieb, durchgeführt werden muss oder ob eine hinreichend lange Auszeit möglich ist.

- Ist eine entsprechende Auszeit möglich, kann innerhalb der bestehenden Datenbank ein neuer LM-Tablespace angelegt werden, in den über Kommandos der Form `ALTER TABLE … MOVE TABLESPACE newts` die Tabellen dieses Tablespaces verschoben werden. Indizes können einfach in einem neuen LM-Tablespace neu aufgebaut werden; dies geht sogar im laufenden Betrieb über den Befehl `ALTER INDEX … REBUILD TABLESPACE newts ONLINE`.
- Ist keine entsprechende Auszeit möglich, kann auch mit den in Oracle und von Drittherstellern zur Verfügung stehenden Reorganisationsmethoden (z.B. `dbms_redefinition`) gearbeitet werden. Hierbei werden die Tabellen im laufenden Betrieb reorganisiert.
- Je nachdem, ob nur ein Tablespace oder die gesamte Datenbank betroffen ist, muss alternativ darüber nachgedacht werden, eine neue Datenbank aufzubauen und diese aus der aktuellen Datenbank zu füllen. Um die Zeitdauer hierbei zu minimieren, gibt es Software, wie z.B. SharePlex von der Firma Quest Software, die hierbei unterstützen kann.

Das in der Datenbank enthaltene Package `dbms_space_admin` bietet im Rahmen der Prozedur `tablespace_migrate_to_local` scheinbar eine weitere einfache Möglichkeit, einen DM-Tablespace direkt in einen LM-Tablespace zu migrieren. Der Tablespace wird dabei lediglich umetikettiert, und die Extent-Bitmap wird hinzugefügt. Die Migration ist im laufenden Betrieb möglich, lediglich umfangreiche Lade-Operationen, die zur Reservierung neuer Extents führen, werden blockiert. Dennoch ist diese Variante in den meisten Fällen nicht praxistauglich, da sie nur eine Umwandlung in einen LM-Tablespace mit einheitlicher Extent-Größe (`UNIFORM SIZE`) zulässt, wobei als Extent-Größe der größte gemeinsame Teiler aller im DM-Tablespace vorhandenen Extent-Größen genommen wird. Dies liegt meistens sehr nahe an der Oracle-Blockgröße und ist somit viel zu klein. Nur für den Fall, dass der DM-Tablespace mit einer MINIMUM SIZE-Storage-Option konfiguriert war, kann die Migration zu einem guten Ergebnis kommen.

3.10.2 Rollback-Segmente

Bis Oracle8i musste die Undo-Verwaltung auf manuellem Wege durchgeführt werden. Bei Erstellung der Datenbank oder unmittelbar danach wurde eine Anzahl Rollback-Segmente erzeugt. Diese manuell erzeugten Rollback-Segmente sind von den automatisch erzeugten Undo-Segmenten (siehe Kapitel 3.3.5) zu unterscheiden. Die manuelle Erstellung geschieht durch Kommandos der Form:

```
SQL> CREATE ROLLBACK SEGMENT r01
     TABLESPACE rbs;
```

Der Tablespace `rbs` ist dabei ein ganz normaler Tablespace.

Wesentliche Nachteile dieser Art der Undo-Verwaltung sind:

- Das Anlegen, die Bestimmung einer geeigneten Größe sowie das Löschen von Rollback-Segmenten muss manuell durch den DBA geschehen
- Insbesondere muss ein geeigneter Wert für den so genannten `OPTIMAL`-Parameter gefunden werden.

- Um Fehlermeldungen der Art `ORA-01555` (Snapshot too old) möglichst zu unterdrücken, müssen Anzahl, Größe und `OPTIMAL`-Parameter der Rollback-Segmente sorgfältig austariert werden.
- Eine Transaktion kann sich nicht über mehrere Rollback-Segmente erstrecken. Für sehr umfangreiche Transaktionen im Rahmen von Batch-Verarbeitungen muss ein dediziertes Rollback-Segment angelegt und in den Skripten kodiert werden.
- Die ab Oracle9i eingeführten Flashback-Funktionen sind mit Rollback-Segmenten nicht nutzbar.

Es ist daher sehr zu empfehlen, ab Oracle9i die automatische Undo-Verwaltung zu benutzen. Die Migration von manueller zu automatischer Undo-Verwaltung ist relativ einfach. Eine kurze Auszeit wird benötigt, da die Datenbank einmal durchgestartet werden muss. Folgendes Verfahren kann dafür genutzt werden (dabei wird davon ausgegangen, dass eine Serverparameterdatei benutzt wird; andernfalls müssen die Serverparameter durch Editieren der Parameterdatei geändert werden):

- Zunächst sollte ein Undo-Tablespace angelegt werden, z.B.:

```
SQL> CREATE UNDO TABLESPACE undotbs
     DATAFILE '…' SIZE 1000M
     AUTOEXTEND ON NEXT 100M MAXSIZE 5000M;
```

- Für den nächsten Neustart der Instanz wird die automatische Undo-Verwaltung eingestellt. Der Parameter `undo_retention` gibt die Zeit in Sekunden an, für die Undo-Informationen aufbewahrt werden sollen, um Lesekonsistenz für Abfragen zu gewährleisten. Der Parameter `undo_tablespace` enthält den Namen des soeben erstellten Tablespaces. Mit `undo_management` wird jetzt auf die automatische Undo-Verwaltung umgeschaltet.

```
SQL> ALTER SYSTEM SET undo_management=AUTO SCOPE=SPFILE;
SQL> ALTER SYSTEM SET undo_tablespace=undotbs SCOPE=SPFILE;
SQL> ALTER SYSTEM SET undo_retention=1800 SCOPE=SPFILE;
SQL> -- den folgenden Befehl nur unter Oracle9i
SQL> ALTER SYSTEM SET undo_suppress_errors=TRUE SCOPE=SPFILE;
```

- Nach dem nächsten Neustart der Instanz arbeitet die Datenbank mit automatischer Undo-Verwaltung. Der ursprüngliche Rollback Tablespace kann jetzt gelöscht werden. Außerdem sollten Sie jetzt auf jeden Fall ein Backup Ihrer Datenbank und der Parameterdatei durchführen.