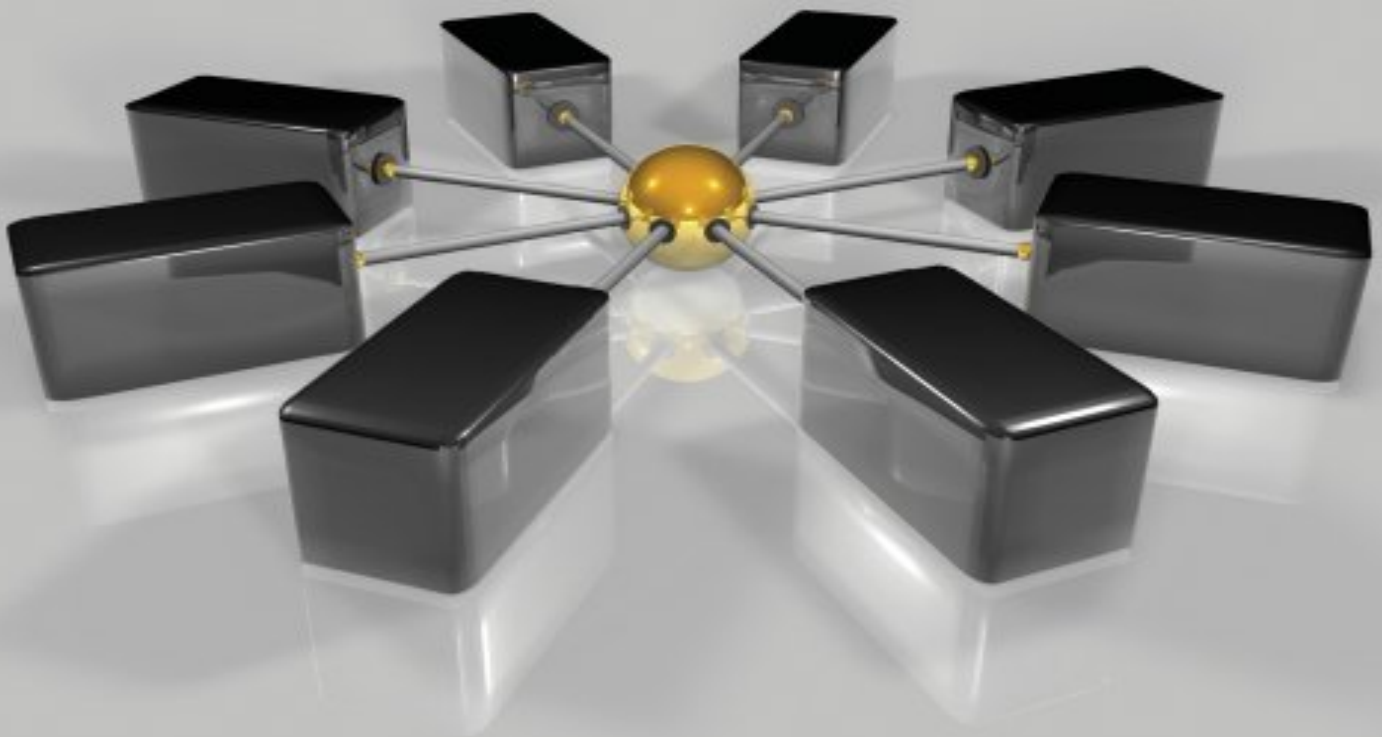


ORACLE®

# Recovery

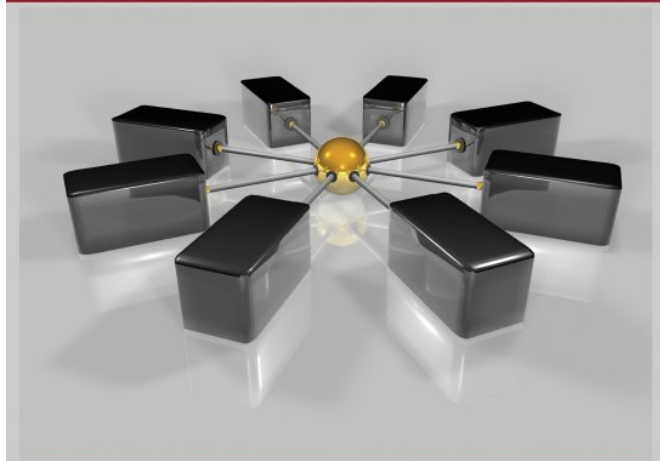
kompakt **Manager**



Eine strukturierte Einführung in den  
Recovery Manager

Marek Adar

ORACLE®  
**Recovery**  
kompakt **Manager**



Eine strukturierte Einführung in den  
Recovery Manager

Marek Adar

ORACLE®

# Recovery Manager Kompakt

Eine strukturierte Einführung in den  
Recovery Manager

Marek Adar

Books on Demand

***Impressum:***

Büren, im August 2011

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über [dnb.d-nb.de](http://dnb.d-nb.de) abrufbar.

© 2011 Marek Adar / Adar-Consult  
Version 3.0.0

Co-Autor: Dennis Andrick / Jeanne Grieger / Dieter Dyhr

Cover-Design: Michael Pastofski

Herstellung und Verlag: Books on Demand GmbH,  
Norderstedt

ISBN 978-3-8448-6757-2

Bildnachweis: © [iStockphoto.com/visual7](http://iStockphoto.com/visual7)

***Für:  
Gaby, Deborah, Ilan und Julian***

## **Im Gedenken an Christa**

*„Hoffnung ist nicht die Überzeugung, dass etwas gut ausgeht, sondern die Gewissheit, dass etwas Sinn hat, egal wie es ausgeht.“*

Vaclav Havel

# **Inhalt**

## **1. Einführung**

## **2. Überblick über den Recovery Manager**

## **3. Die Oracle-Datenbankarchitektur**

### ***3.1. Die Instanz und die Datenbank***

### ***3.2. Die Oracle-Instanz***

### ***3.3. Die Oracle-Datenbank***

### ***3.4. Die Kontrolldatei***

### ***3.5. Transaktionen***

### ***3.6. Die System Global Area***

### ***3.7. Der Database-Buffer-Cache***

### ***3.8. Die Dirty-List***

### ***3.9. Der Redo-Log-Buffer***

### ***3.10. Der Redo-Log-Writer***

### ***3.11. Die Redo-Log-Dateien***

### ***3.12. Der Checkpoint-Prozess und der Database-Writer***

### ***3.13. Warum der Umweg über die Redo-Log-Dateien?***

### ***3.14. Undo-Segmente und Lesekonsistenz***

### ***3.15. Instanz-Recovery***

### ***3.16. Der Shared Pool***

### ***3.17. Weitere Speicherbereiche in der SGA***

#### ***3.17.1. Der Large Pool***

- 3.17.2. Der Java Pool
- 3.17.3. Der Streams Pool

### ***3.18. Zusammenfassung***

## **4. Die Flash Recovery Area**

### ***4.1. Konfiguration der Flash Recovery Area***

### ***4.2. Informationen zur Flash Recovery Area***

### ***4.3. Zusammenfassung***

### ***4.4. Auf einen Blick***

- 4.4.1. Parameter
- 4.4.2. Views

## **5. Der ARCHIVELOG-Modus**

### ***5.1. Archivierungsziele***

### ***5.2. Archivformat***

### ***5.3. Automatisches Starten des Archivierungsprozesses***

### ***5.4. Aktivierung des ARCHIVELOG-Modus***

### ***5.5. Überprüfung des ARCHIVELOG-Modus***

### ***5.6. Zusammenfassung***

### ***5.7. Auf einen Blick***

- 5.7.1. Parameter
- 5.7.2. Aktivierung des ARCHIVELOG-Modus
- 5.7.3. Views

## **6. Architektur des Recovery Managers**

### ***6.1. Anmelden am Recovery Manager***

### ***6.2. Bedienung des Recovery Managers***

- 6.2.1. Interaktiver Modus



6.2.2. Batch Modus

**6.3. Befehlsarten des Recovery Managers**

**6.4. SQL aus dem Recovery Manager**

**6.5. Zusammenfassung**

**6.6. Auf einen Blick**

## **7. Konfiguration des Recovery Managers**

**7.1. Konfiguration einer Erhaltungsrichtlinie**

7.1.1. Redundanz

7.1.2. Wiederherstellungsfenster

7.1.3. Löschen veralteter Sicherungen

7.1.4. Deaktivieren einer Erhaltungsrichtlinie

**7.2. Sicherungsoptimierung**

**7.3. Ausschließen von Tablespaces aus der Sicherung**

**7.4. Automatische Sicherung der Kontroll- und Serverparameterdatei**

**7.5. Löschen und Zurücksetzen der Konfiguration**

**7.6. Zusammenfassung**

**7.7. Auf einen Blick**

## **8. Durchführen von Datenbanksicherungen**

**8.1. Der Sicherungstyp**

8.1.1. Backupsets

8.1.2. Image-Kopien

8.1.3. Vorkonfiguration des Sicherungstyps

**8.2. Der Backup-Befehl**

**8.3. Sicherungskanäle**

8.3.1. Manuelle Kanalerstellung

8.3.2. Sicherung auf Band

8.3.3. Sicherung auf Platte

#### ***8.4. Backup-Pieces***

#### ***8.5. Sicherungsbezeichner***

#### ***8.6. Erstellen von Sicherungskopien***

#### ***8.7. Verwenden mehrerer Sicherungsprozesse***

8.7.1. Vorkonfiguration mehrerer Sicherungsprozesse

8.7.2. Manuelle Erzeugung mehrerer Sicherungsprozesse

#### ***8.8. Differenzielle inkrementelle Sicherungsstrategien***

8.8.1. Ebenen von differenziellen inkrementellen Sicherungen

8.8.2. Kumulative inkrementelle Sicherungen

#### ***8.9. Block Change Tracking***

8.9.1. Aktivieren des Block Change Trackings

8.9.2. Überwachen von Block Change Tracking

#### ***8.10. Inkrementell aktualisierte Sicherungen***

8.10.1. Image-Kopien und differenzielle Sicherungen

8.10.2. Anwenden der differenziellen Sicherungen

#### ***8.11. Sichern von Archiven***

8.11.1. Sichern aller Archive

8.11.2. Sichern der Archive ab einer Sequenznummer

8.11.3. Sichern von Archiven mit LIKE

8.11.4. Erstellen mehrerer Sicherungskopien der Archive

8.11.5. Löschen von Archiven nach der Sicherung

### ***8.12. Sichern der Kontrolldatei***

### ***8.13. Sichern der gesamten Datenbank und Archive***

### ***8.14. Sichern von Disk-Sicherungen auf Band***

### ***8.15. Zusammenfassung***

### ***8.16. Auf einen Blick***

8.16.1. Sicherungsbefehle

8.16.2. Platzhalter für die Formatierung von Sicherungsdateien

8.16.3. Vorkonfigurationen

8.16.4. Views

## **9. Verwaltung des Sicherungskatalogs**

### ***9.1. Erstellung einer Sicherungskatalogdatenbank***

9.1.1. Durchführung der Erstellung einer Sicherungskatalogdatenbank

### ***9.2. Registrieren von Datenbanken im Sicherungskatalog***

### ***9.3. Deregistrieren von Datenbanken aus dem Sicherungskatalog***

### ***9.4. Synchronisation des Sicherungskatalogs***

### ***9.5. Katalog-Upgrade für Datenbanken***

### ***9.6. Auflisten der erstellten Sicherungen mit LIST***

9.6.1. Anzeigen von Backupset-Sicherungen

9.6.2. Anzeigen von Image-Kopien

### ***9.7. Sicherungsinformation mit REPORT***

9.7.1. Anzeigen der Datenbankstruktur

9.7.2. Anzeigen veralteter Sicherungen

9.7.3. Anzeigen der Sicherungsnotwendigkeit

### ***9.8. Löschen von Sicherungen***

9.8.1. Löschen veralteter Sicherungen

9.8.2. Löschen spezifischer Backupsets

9.8.3. Löschen spezifischer Kopien

9.8.4. Löschen von Archiven

9.8.5. Die Option FORCE

9.8.6. Verwenden der Option NOPROMPT

9.8.7. Überprüfung des Katalogs mit  
CROSSCHECK

### ***9.9. Statusänderung von Backupsets und Kopien mit CHANGE***

9.9.1. Backupsets und Kopien als nicht verfügbar  
markieren

9.9.2. Langzeitsicherungen

### ***9.10. Sicherungen katalogisieren mit CATALOG***

### ***9.11. Gespeicherte Sicherungsskripte***

9.11.1. Erstellen von gespeicherten  
Sicherungsskripten

9.11.2. Ändern und Löschen von gespeicherten  
Sicherungsskripten

9.11.3. Anzeigen von gespeicherten Skripten

9.11.4. Ausführen von gespeicherten  
Sicherungsskripten

### ***9.12. Sichern der Katalogdatenbank***

9.12.1. Sicherungsstrategien für den Sicherungskatalog

9.12.2. Export/Import des Sicherungskatalogs

### ***9.13. Sicherungskatalog-Views***

9.13.1. Wichtige Katalog-Views

9.13.2. Beispiele zur Verwendung der Katalog-Views

### ***9.14. Zusammenfassung***

### ***9.15. Auf einen Blick***

## **10. Wiederherstellung von Datenbanken**

### ***10.1. Theorie der Wiederherstellung***

10.1.1. Warum ist die Konsistenz so wichtig?

### ***10.2. RESTORE und RECOVER***

### ***10.3. Wiederherstellung im NOARCHIVELOG-Modus***

### ***10.4. Vollständige Wiederherstellung im ARCHIVELOG-Modus***

10.4.1. Grundregeln der Wiederherstellung

10.4.2. Informationen über defekte Dateien

10.4.3. Wiederherstellung nicht kritischer Datendateien im ARCHIVELOG-Modus

10.4.4. Wiederherstellung von systemkritischen Datendateien im ARCHIVELOG-Modus

10.4.5. Verwendung von SET NEWNAME

10.4.6. Wiederherstellung einer Datenbank über inkrementell aktualisierte Sicherungen und Image-Kopien

### ***10.5. Unvollständige Wiederherstellung***

- 10.5.1. Theorie der unvollständigen Wiederherstellung
- 10.5.2. Arten der unvollständigen Wiederherstellung
- 10.5.3. Zeitbasierte Wiederherstellung
- 10.5.4. SCN-basierte unvollständige Wiederherstellung
- 10.5.5. Sequenz-basierte unvollständige Wiederherstellung
- 10.5.6. Unvollständige Wiederherstellung vor RESETLOGS

## ***10.6. Wiederherstellung der Kontrolldateien***

- 10.6.1. Wiederherstellung der Kontrolldateien aus dem AUTOBACKUP
- 10.6.2. Wiederherstellung der Kontrolldateien aus einem Backupset ohne Sicherungskatalogdatenbank und AUTOBACKUP
- 10.6.3. Wiederherstellung der Kontrolldateien mit einer Sicherungskatalogdatenbank

## ***10.7. Disaster Recovery***

- 10.7.1. Schritte des Disaster Recovery
- 10.7.2. Disaster Recovery einer Datenbank mit einer Sicherungskatalogdatenbank
- 10.7.3. Disaster Recovery einer Datenbank ohne Sicherungskatalogdatenbank

## ***10.8. Klonen einer Datenbank***

- 10.8.1. Theorie des Klonens
- 10.8.2. Der Befehl DUPLICATE
- 10.8.3. Schritte des Klonens
- 10.8.4. Klonen einer Datenbank unter Verwendung einer Sicherungskatalogdatenbank

10.8.5. Klonen einer Datenbank ohne Sicherungskatalogdatenbank

10.8.6. Klonen einer Datenbank ohne Backup

## ***10.9. BLOCKRECOVER***

10.9.1. Erkennen von defekten Blöcken

10.9.2. Der Befehl BLOCKRECOVER

## ***10.10. Der Recovery Advisor***

10.10.1. LIST FAILURE

10.10.2. ADVISE FAILURE

10.10.3. REPAIR FAILURE

10.10.4. CHANGE FAILURE

10.10.5. Beispiel: Verlust des System-Tablespace

10.10.6. Beispiel: Verlust der Kontrolldateien

## ***10.11. Flashback-Database***

10.11.1. Aktivierung der Flashback-Database

10.11.2. Informationen über die Flashback-Database

10.11.3. Zurücksetzen einer Flashback-Database

10.11.4. Durchführung von Flashback

10.11.5. Verwenden von Wiederherstellungspunkten

10.11.6. Garantierte Wiederherstellungspunkte

10.11.7. Einschränkungen der Flashback-Database

## ***10.12. Zusammenfassung***

## ***10.13. Alles auf einen Blick***

10.13.1. Vollständige Wiederherstellung

10.13.2. Unvollständige Wiederherstellung

10.13.3. Wiederherstellen der Kontrolldatei

10.13.4. Klonen einer Datenbank

10.13.5. Blockrecover  
10.13.6. Recovery Advisor  
10.13.7. Flashback-Database

**Stichwortverzeichnis**

**Abbildungsverzeichnis**

**Danksagung**

**Haftungshinweis**

**Linksammlung**

**Weitere interessante Publikationen**

**Unternehmen für Oracle Schulungen und  
Consulting**



# 1. Einführung

Liebe Leserin, lieber Leser,

viele Oracle-Datenbanken werden heute noch mit konventionellen Mitteln, beispielsweise durch Herunterfahren der Instanz und Kopieren der Datenbankdateien, gesichert. Diese Sicherungsstrategie erfolgt hauptsächlich deshalb, weil das grundlegende Wissen für die Verwendung des Recovery Managers fehlt.

Diese kompakte Einführung soll das grundlegende Wissen für die Verwendung des Recovery Managers vermitteln und mit seinen wichtigen Features vertraut machen. Aufgrund der aktuellen Funktionalitäten des Recovery Managers ist es sinnvoll, ihn für die Durchführung von Sicherungen in den jetzigen Versionen von Oracle zu verwenden.

Dieses Buch erhebt nicht den Anspruch auf Vollständigkeit, sondern bietet eine strukturierte Einführung in die Handhabung des Recovery Managers, die als Basis für die komplexere Verwendung dienen soll.

Für das strukturierte Aufsetzen und Wiederherstellen von Datenbanksicherungen ist grundlegendes Wissen über die Oracle-Architektur Voraussetzung. Die Arbeitsweise der Architektur wird in allen grundlegenden Funktionsweisen der Komponenten in diesem Buch angesprochen. Die Architektur ist in der Realität noch detaillierter als hier beschrieben, ist aber für das Verstehen einer Durchführung von Sicherungen mithilfe des Recovery Managers ausreichend.

In dieser Einführung wurde bewusst auf die Verwendung des Enterprise Managers von Oracle verzichtet, da das Verstehen der Befehlssyntax des Recovery Managers die Grundlage und vorrangig für das Durchführen von

Sicherungen ist - gegenüber dem Umgang mit der grafischen Oberfläche.

Ist die Befehlssyntax verstanden, so steht der Verwendung der grafischen Oberfläche nichts mehr im Wege. Eine Fokussierung nur auf den Enterprise Manager ist für die Verwendung des Recovery Managers nicht dienlich, da eine Vielzahl von Softwareherstellern die Sicherungswerkzeuge für Oracle herstellen, den Recovery Manager verwenden und ihn von außen mit den hier angesprochenen Befehlen versorgen. Um zu verstehen, in welcher Art und Weise diese Sicherungswerkzeuge den Recovery Manager verwenden, wird der Fokus auf die Befehlssyntax gelegt.

Die Befehlssyntax selber ist nicht schwer zu erlernen, wenn die Philosophie hinter der Struktur der Syntax verinnerlicht wurde.

Die Erfahrung bei der Vermittlung des Recovery Managers in meinen Schulungen hat gezeigt, dass dieser Weg, den Recovery Manager nahe zu bringen, erfolgreich ist und die bei vielen Administratoren vorhandenen Ängste abgebaut hat.

Sollten Sie Anregungen, Fragen oder Verbesserungsvorschläge zu diesem Buch haben, würde ich mich sehr über eine Mail von Ihnen freuen. Senden Sie einfach Ihr Anliegen an [info@adar-consult.de](mailto:info@adar-consult.de).

Und nun viel Spaß beim Lesen und Ausprobieren.

Herzlichst, Ihr



## 2. Überblick über den Recovery Manager

Die Einführung des Recovery Managers, der seitdem stetig weiterentwickelt wurde, erfolgte in Oracle, Version 8. Der Recovery Manager ist ein Kommandozeilenwerkzeug, welches einen eigenen Befehlssatz zur Sicherung von Oracle-Datenbanken sowie zur Wartung der erstellten Sicherungen verwendet. Von Release zu Release erhielt der Recovery Manager immer weitere Funktionalitäten, die die Durchführung von Datenbanksicherungen immer komfortabler gestalteten. Bis zur Version 11g besitzt der Recovery Manager heute folgende Funktionalitäten:

- Datensicherung auf Band, Platte und Flash Recovery Area
- Komprimierung von Sicherungen
- Verschlüsselung von Sicherungen
- Differenzielle Sicherungen
- Sicherung als Backupset
- Sicherung als Image-Kopie
- Blockprüfung
- Block Recovery
- Inkrementelles Recovery von Image-Kopien
- Wiederherstellung von Datenbanken und Dateien
- Unvollständiges Recovery
- Flashback
- Sicherungskatalogwartung
- Duplizieren von Datenbanken
- Transportable Tablespace
- Transportable Database
- Recovery Advisor

Aufgrund dieser Funktionalitäten gibt es heute keinen Grund mehr, Oracle-Datenbanken mit anderen Techniken zu sichern.

***Ein Beispiel:***

Der Recovery Manager führt während der Sicherung eine Überprüfung der Datenbankblöcke durch, womit eine Blocksicherheit garantiert ist. Zusätzlich inventarisiert er alle Metadaten der durchgeführten Sicherungen in einem eigenen Sicherungskatalog, der im Fehlerfall die Wiederherstellung der Datenbank vereinfacht.

Für die Implementierung einer Sicherungsstrategie sowie die Wiederherstellung der gesamten Datenbank oder Teilen davon ist die Kenntnis der Datenbankarchitektur von enormer Wichtigkeit. Ansonsten kann der Wiederherstellungsprozess nicht verstanden werden.

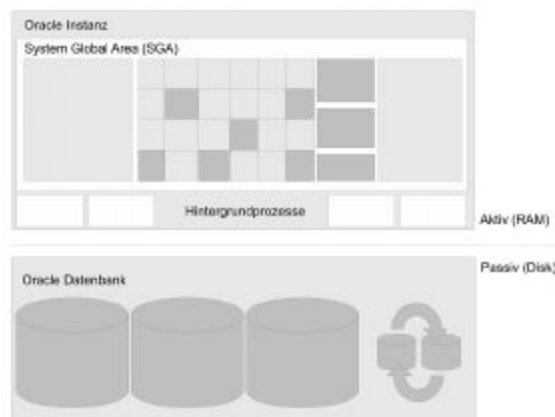
## 3. Die Oracle-Datenbankarchitektur

Jeder Anwender hat eine gewisse Erwartung an eine Datenbank, die sich in der Regel mit folgenden Schlagworten umschreiben lässt: Eine schnelle Antwortzeit, eine optimale Schreibgeschwindigkeit und Datensicherheit. Diese grundlegenden Erwartungen sollen in diesem Kapitel in einer Zusammenfassung der Datenbankarchitektur wiedergegeben und durch die grundlegenden Komponenten der Architektur beschrieben werden, um ein Gesamtbild der Funktionsweise vom Oracle-RDBMS zu bekommen.

### 3.1. Die Instanz und die Datenbank

Die Oracle-Architektur gliedert sich grob in zwei grundlegende Bereiche:

- die Oracle-Datenbank
- die Oracle-Instanz



**Abbildung 1. Oracle-Architektur**

### 3.2. Die Oracle-Instanz

Die Oracle-Instanz ist der Motor der Oracle-Architektur und befindet sich im Hauptspeicher des Systems. Die Oracle-Instanz wird auch als der **aktive Teil** der Oracle-Architektur bezeichnet. Zu ihr gehören Speicherstrukturen für die

Ablage von Daten sowie Hintergrundprozesse. Der Aufbau der Instanz wird über die Parameter- oder (ab Oracle 9i) über die Serverparameterdatei definiert. Wird eine Instanz gestartet, so wird im Vorfeld der Inhalt der Parameterdatei oder der Serverparameterdatei ausgelesen, die die Konfigurationseinstellungen der Instanz beinhaltet. So werden beispielsweise die Größen der Speicherstrukturen, aber auch bestimmte Optionsparameter der Datenbank und Instanz über sie konfiguriert.

### ***3.3. Die Oracle-Datenbank***

Die Oracle-Datenbank besteht aus den Datenbankdateien und befindet sich auf dem Plattensubsystem des Datenbankservers. Die Oracle-Datenbank wird auch als der **passive Teil** bezeichnet. Allgemein wird gesagt, dass eine Datenbank gestartet wird. Dies ist aber nicht richtig, weil nur der Motor, also die Instanz, gestartet werden kann, welche dann mit der Datenbank interagiert.

### ***3.4. Die Kontrolldatei***

Die Kontrolldatei ist ein wichtiger Bestandteil der Oracle-Datenbank. In ihr befinden sich unter anderem die Speicherorte der Datenbankdateien. Nach dem Start der Instanz wird die Kontrolldatei über den in der Parameterdatei befindlichen Initialisierungsparameter `CONTROL_FILES` lokalisiert und die Speicherorte der Datenbankdateien werden ausgelesen. Darauffolgend werden die Datenbankdateien an die Instanz angebunden (gemountet). Ist die Kontrolldatei defekt oder nicht vorhanden, schlägt ein Öffnen der Datenbank fehl, weil die entsprechenden Datendateien nicht gefunden werden können.

Zusätzlich wird die Kontrolldatei vom Recovery Manager als Sicherungskatalog verwendet, indem alle Metadaten der mit

dem Recovery Manager erzeugten Sicherungen in ihr gespeichert werden. Ist die Kontrolldatei unwiederbringlich verloren, kann die Datenbank nur schwer wiederhergestellt werden. Aus diesem Grund ist eine Spiegelung der Kontrolldateien zu empfehlen, um einem Verlust vorzubeugen.

### **3.5. Transaktionen**

Transaktionen sind ein wichtiger Bestandteil einer Datenbank, die abhängige Datenänderungen zusammenfassen. Dies bedeutet, dass diese Änderungen nicht voneinander getrennt werden dürfen und alle erfolgreich ausgeführt werden müssen. Sollten eine oder mehrere Änderungen in einer Transaktion nicht erfolgreich durchgeführt werden, darf auch keine andere Änderung dieser Transaktion durchgeführt werden.

#### **Beispiel:**

Bei einer Überweisung wird Geld von einem Konto auf ein anderes transferiert. Das bedeutet, dass Geld von einem Konto abgebucht und einem anderen Konto gutgeschrieben wird. Grob betrachtet, sind dies zwei Änderungen, denn von dem ersten Konto wird der Geldbetrag abgezogen, auf das zweite Konto wird dieser Betrag addiert. Diese beiden Änderungen sind unmittelbar miteinander verknüpft und voneinander abhängig. Sollte nun aber die Abbuchung des Geldbetrages erfolgreich sein, das Gutschreiben aber fehlschlagen, so ist Geld vernichtet worden. Im umgekehrten Fall gilt: Schlägt die Abbuchung fehl, während das Gutschreiben erfolgreich ist, so ist Geld vermehrt worden. In beiden Fällen ist die Konsistenz der Datenbank gestört, weil nicht nachvollzogen werden kann, wohin der entsprechende Geldbetrag gegangen ist bzw. wo er herkommt.

Notwendigerweise müssen diese beiden Änderungen als Einheit betrachtet und in einer Transaktion zusammengefasst werden. Unter Oracle werden Transaktionen mit der Anmeldung und der ersten Änderung von Daten gestartet. Folgen weitere Änderungen, gehören diese mit zur aktuell laufenden Transaktion. Diese Transaktion kann bei erfolgreicher Durchführung der Änderung mit dem Befehl COMMIT festgeschrieben, bei nicht erfolgreicher Durchführung einzelner Änderungen mit dem Befehl ROLLBACK rückgängig gemacht werden. Nach dem Festschreiben oder Zurückführen der Transaktion startet automatisch eine neue Transaktion bei einer erneuten Änderung von Daten.

*Beispiel einer Transaktion:*

```
SQL> connect ac
Kennwort eingeben:
Connect durchgeführt.

SQL> SELECT KUNNR, NACHNAME, VORNAME FROM KUNDEN;

  KUNNR NACHNAME          VORNAME
-----
  2124 Meier             Hans
  4711 Schulz           Willy

SQL> UPDATE KUNDEN SET NACHNAME='Muller' WHERE KUNNR=2124;
1 Zeile wurde aktualisiert.

SQL> UPDATE KUNDEN SET NACHNAME='Muller' WHERE KUNNR=4711;
1 Zeile wurde aktualisiert.

SQL> SELECT KUNNR, NACHNAME, VORNAME FROM KUNDEN;

  KUNNR NACHNAME          VORNAME
-----
  2124 Muller            Hans
  4711 Muller           Willy

SQL> rollback;

Transaktion mit ROLLBACK rückgängig gemacht.

SQL> SELECT KUNNR, NACHNAME, VORNAME FROM KUNDEN;

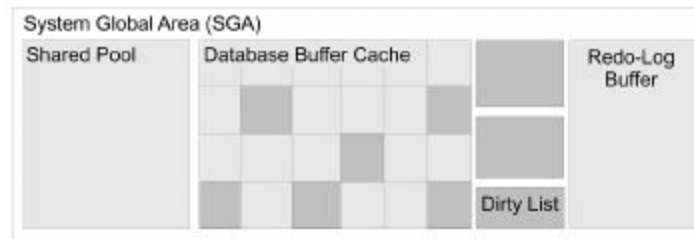
  KUNNR NACHNAME          VORNAME
-----
  2124 Meier             Hans
  4711 Schulz           Willy
```

### ***3.6. Die System Global Area***

Die **S**ystem **G**lobal **A**rea (SGA) beinhaltet die Speicherstrukturen der Oracle-Instanz, welche unter anderem Tabellendaten, Metadaten oder Systeminformationen der Datenbank speichern. Zu den



Speicherstrukturen gehören zum Beispiel der Database-Buffer-Cache, der Redo-Log-Buffer, der Shared Pool, der Large Pool, der Java Pool und weitere hier nicht näher erläuterte Speicherbereiche.



**Abbildung 2. System Global Area (SGA)**

### **3.7. Der Database-Buffer-Cache**

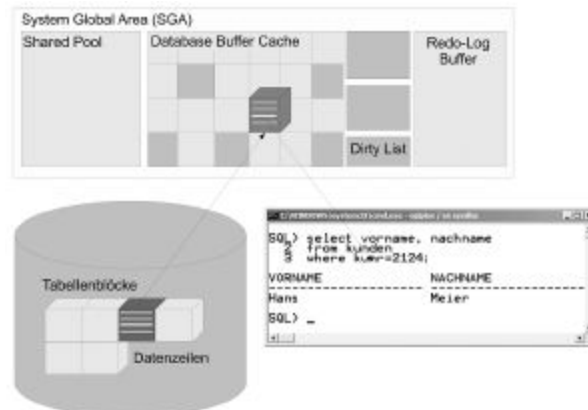
Der Database-Buffer-Cache speichert Datenblöcke der Datenbank. Die kleinste zu lesende oder zu schreibende Einheit der Oracle-Datenbank ist der Oracle-Datenbankblock. In den Datenbankblöcken werden beispielsweise Zeilen von Tabellen abgelegt.

Wird über eine SQL-Abfrage ein Datensatz in einer Tabelle gesucht, so wird der Tabellenblock, in dem sich diese Zeile befindet, in den Database-Buffer-Cache geladen, aber nur die gesuchte Zeile angezeigt.

Durch das Laden des gesamten Blocks werden alle Datensätze, die sich in diesem Block befinden, mit geladen. Sollte der gleiche oder ein anderer Anwender Daten benötigen, die sich ebenfalls in diesem Block befinden, so fällt ein erneutes Laden des Blockes vom Plattensystem weg, wodurch eine erhöhte Zugriffsgeschwindigkeit erreicht wird. Durch eine optimale Größenkonfiguration des Database-Buffer-Caches können alle wichtigen Bewegungsdaten der Datenbank im Hauptspeicher vorgehalten werden.

Damit Oracle erkennen kann, welche Blöcke mehr benötigt werden als andere, wird die Präsenz der Blöcke im

Database-Buffer-Cache durch einen LRU-Algorithmus (**L**east **R**ecently **U**sed) verwaltet. Das bedeutet, dass die Blöcke, auf die häufig zugegriffen wird, länger im Hauptspeicher vorgehalten werden als jene, die nur einmalig geladen und danach nicht mehr benötigt werden.



**Abbildung 3. Laden von Blöcken in den Database-Buffer-Cache**

Durch diesen LRU-Algorithmus fallen dann nicht mehr benötigte Blöcke aus dem Database-Buffer-Cache heraus, um Platz für neue zu schaffen. Die Blockgröße wird während der Installation einer Datenbank festgelegt und kann zwischen 2 und 32 Kilobyte liegen. Eine Änderung der Blockgröße ist nach der Installation einer Datenbank nicht mehr möglich und kann nur durch eine Neuinstallation erreicht werden. Die eingestellte Standardblockgröße beträgt 8 KB.

Abhängig von der beabsichtigten Betriebsart einer Datenbank sollte die Blockgröße gesetzt werden: In sogenannten OLTP-Datenbanken (**O**n-**L**ine **T**ransaction **P**rocessing) sind in der Regel kleinere Blöcke, in OLAP-Datenbanken (**O**n-**L**ine **A**nalytical **P**rocessing) größere Blöcke zu bevorzugen.

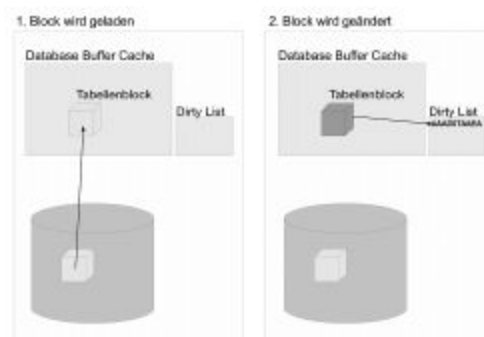
**OLTP-Datenbanken** zeichnen sich durch eine hohe Transaktionsrate aus, deren Datenänderungen innerhalb der Transaktionen klein sind. Zusätzlich laufen viele Abfragen in die Datenbank ein, deren Ergebnismengen klein sind. Bei

diesen Datenbanken werden meist kleine Blockgrößen bevorzugt, um pro Datensatzszugriff wenige Datensätze zusätzlich pro Block laden zu müssen. OLTP-Systeme sind zum Beispiel ERP-Systeme (Enterprise Resource Planning, Personalplanung, Kapital, Betriebsmittel, Verkauf, Marketing, Finanz- und Rechnungswesen) oder CRM-Systeme (Customer Relation Management, Systeme für Kundenbetreuung).

**OLAP-Datenbanken** werden in bestimmten Abständen mit Daten befüllt und dienen zur Analyse dieser Datenbestände. Da in diesen Datenbanken große Datenmengen verarbeitet werden, werden hier größere Blöcke bevorzugt, da mit dem Lesen eines Datensatzes, der sich in einem solchen großen Block befindet, gleichzeitig viele zusätzliche Datensätze geladen und verarbeitet werden können.

### ***3.8. Die Dirty-List***

Werden Datensatzänderungen innerhalb der Datenbank durchgeführt, so werden die Blöcke, in denen die Änderungen vollzogen werden, nicht direkt in die Datenbank zurückgeschrieben. Vielmehr werden die Blockadressen der geänderten Blöcke in einem Speicherbereich, der Dirty-List, protokolliert. Somit befinden sich im Database-Buffer-Cache die geänderten Blöcke, während in der Datenbank die Blöcke noch ihren Originalzustand besitzen.



**Abbildung 4. Verwendung der Dirty-List**

Erst zu späterer Zeit werden sie dann anhand der Dirty-List aus dem Database-Buffer-Cache in die Datenbank übertragen. Im Allgemeinen kann man sagen, dass sich Änderungen primär immer in der Instanz vollziehen und erst später in die Datenbank geschrieben werden, um auf diese Weise einen Geschwindigkeitsvorteil zu erreichen.

### ***3.9. Der Redo-Log-Buffer***

Aufgrund der Tatsache, dass Änderungen erst im Database-Buffer-Cache durchgeführt werden, gingen diese Änderungen bei einem Instanzabsturz verloren. Um dies zu verhindern, werden Änderungen zusätzlich im Redo-Log-Buffer protokolliert. Für jede Datensatzänderung werden unter anderem die Blockadresse, der Ort der Änderung innerhalb des Blockes, der neue Wert, ein Zeitstempel und eine Systemänderungsnummer (**S**ystem **C**hange **N**umber, SCN) in diesen Speicherbereich geschrieben. Unter Oracle erhalten alle Änderungen, die zur gleichen Transaktion gehören, eine Systemänderungsnummer, um alle Änderungen einer Transaktion identifizieren zu können.

Trotz all dieser Informationen, die im Redo-Log-Buffer festgehalten werden, gingen vorgenommene Änderungen im Falle eines Instanzabsturzes verloren, da dieser Speicherbereich in Konsequenz selbst gelöscht würde. Somit muss es einen Mechanismus geben, der Datenverlust im soeben beschriebenen Szenario verhindert. Die einzige Möglichkeit, dieses Problem zu lösen, besteht darin, die Inhalte des Redo-Log-Buffers auf einem Permanentdatenträger zu speichern.

### ***3.10. Der Redo-Log-Writer***

Der Inhalt des Redo-Log-Buffers wird durch einen Hintergrundprozess, den sogenannten Redo-Log-Writer, regelmäßig geleert und in die Redo-Log-Dateien