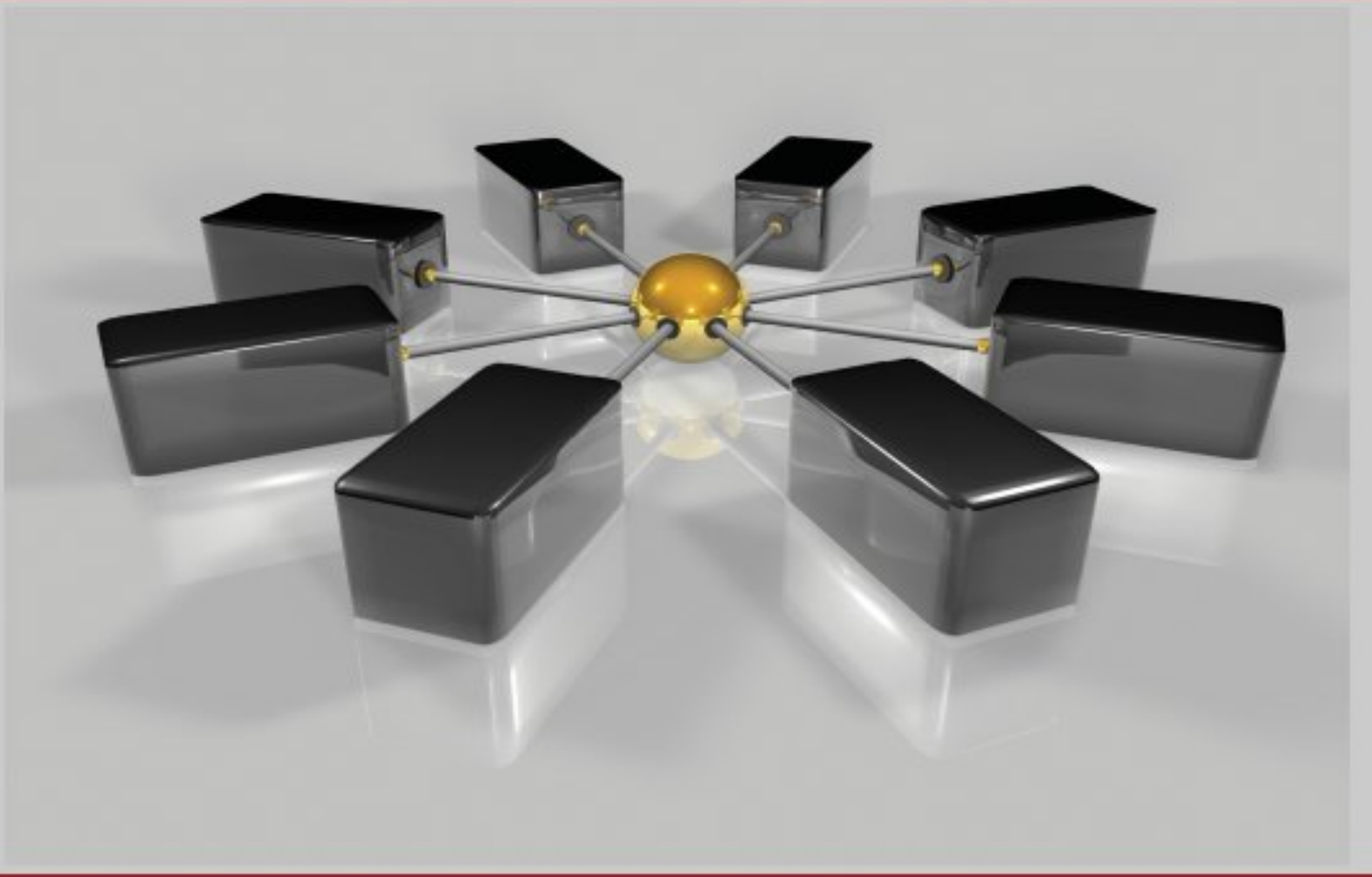
ORACLE®

Recovery kompakt Manager

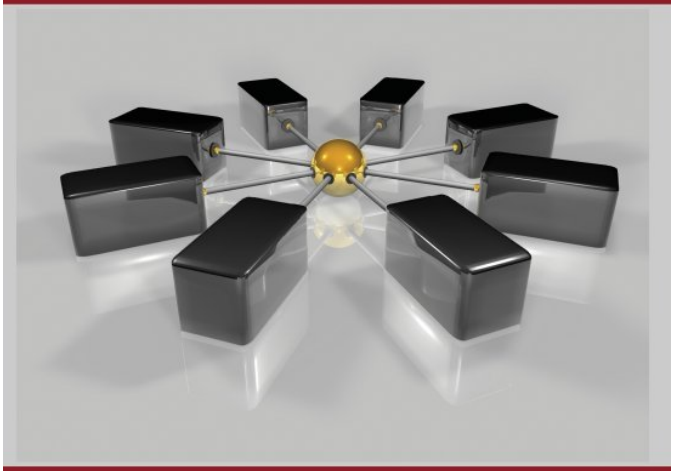


Eine strukturierte Einführung in den Recovery Manager

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 abrufbar.

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

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

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.

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

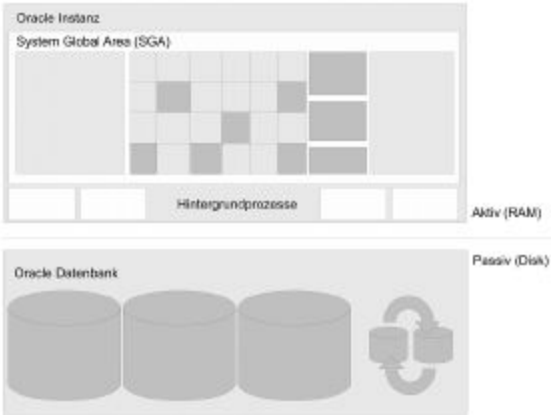


Abbildung 1. Oracle-Architektur

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 wer-den.

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 KUNR, NACHNAME, VORNAME FROM KUNDEN;

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

SQL> UPDATE KUNDEN SET NACHNAME='Müller' WHERE KUNR=2124;

1 Zeile wurde aktualisiert.

SQL> UPDATE KUNDEN SET NACHNAME='Müller' WHERE KUNR=4711;

1 Zeile wurde aktualisiert.

SQL> SELECT KUNR, NACHNAME, VORNAME FROM KUNDEN;

  KUNR NACHNAME              VORNAME
------ --------------------- ----------------------
  2124 Müller                Hans
  4711 Müller                Willy

SQL> rollback;

Transaktion mit ROLLBACK rückgängig gemacht.

SQL> SELECT KUNR, NACHNAME, VORNAME FROM KUNDEN;

  KUNR 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.

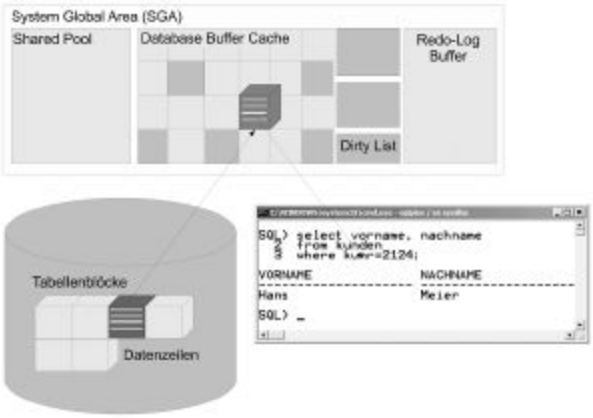


Abbildung 3. Laden von Blöcken in den Database-Buffer-Cache

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 Datensatzsatzzugriff 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.

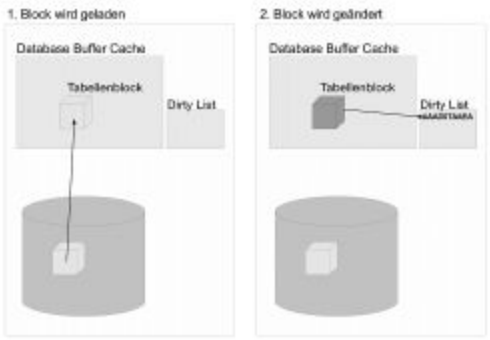


Abbildung 4. Verwendung der Dirty-List

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