Advanced education: Oracle [5] Certified Professional Database Administrator

Stefan Hietel [1], Robert Warnke [2]

16. Juli 2004

Inhaltsverzeichnis

0.1 Intro

0.1.1 Möglichkeiten der Zertifizierung

Examen zum Oracle8i/9i Certified Professional Database Administrator:

- Exam 1Z0-001 Introduction to Oracle: SQL and PL/SQL
- Exam 1Z0-007 Introduction to Oracle9i: SQL
- Exam 1Z0-023 Oracle8i/9i: Architecture and Administration
- Exam 1Z0-025 Oracle8i/9i: Backup and Recovery
- Exam 1Z0-024 Oracle8i/9i: Performance Tuning
- Exam 1Z0-026 Oracle8i/9i: Network Administration

Upgradezertifizierung zum Oracle9i Certified Professional Database Administrator. Sie müssen bereits Oracle 8.i-Administrator sein und folgendes Examen bestehen:

- Exam 1Z0-030 Oracle9i Database: New Features for Administrators

Examen können bei jedem Prometric-Testcenter abgelegt werden. Eine vorherige Anmeldung (48 Stunden vorher) ist notwendig. Die Kosten je Prüfung belaufen sich zum gegenwärtigen Zeitpunkt auf 125 Euro netto (Stand 1.1.2003). Eine Übersicht aller Testcenter finden Sie unter [12]. Hier können Sie sich auch ONLINE registrieren.

Die Prüfungen bestehen größtenteils aus Multiple-Choice-Aufgaben, je Prüfung etwa 50 bis 60 Fragen. Das Passing-Score liegt zwischen 60 und 75 Prozent. Es werden nicht alle Fragen gewertet. Welche Fragen nicht gewertet wurden, erfahren Sie leider nicht.

Alle aktuellen Informationen und Änderungen hinsichtlich der Oracle-Zertifizierungen können unter [6]

Kapitel 1

Einführung in PL / SQL Oracle 8i/9i

1.0.1 Oracle 9i Installation unter Linux

Diese Anleitung gilt für eine Installation für Schulungszwecke.

Schwierigkeiten bei der Installation unter Linux resultieren daraus, dass vor der Installation das Betriebssystem zunächst konfiguriert werden muß (z.B. Konfiguration der Kernelparameter). Desweiteren ist die Installation an sich noch nicht sehr gut automatisiert.

Systemvoraussetzungen

Um Oracle 9i zu installieren, empfiehlt der Hersteller folgende Systemvoraussetzungen:
Arbeitsspeicher: Minimum 512 MB RAM
SWAP Space: Optimal sind 1 GB aber mindestens so groß wie der Arbeitsspeicher.

Download

Oracle Database Enterprise/Standard Edition läßt sich kostenlos nach Registrierung von [7] herunterladen. Die Datenbank besteht aus drei Paketen, die hier nach einer Anmeldung runtergeladen werden können. Folgende Pakete müssen vorliegen:

- lnx_920_disk1.cpio.gz (553,239,173 bytes)
- lnx_920_disk2.cpio.gz (588,798,999 bytes)
- lnx_920_disk3.cpio.gz (442,087,410 bytes)

Außerdem sollte von [11] das Paket `orarun.rpm` runtergeladen werden, das später noch benötigt wird, um die Kernelparameter zu konfigurieren.

Erzeugen der User und Gruppen

Die Installation und Administration von Oracle sollte unter einem gesonderten Nutzer erfolgen. Dieser Nutzer heißt `oracle` und wird von Suse schon bei der Installation der Distribution angelegt. Er gehört zu den Gruppen `dba` und `oinstall` und erhält sein persönliches Verzeichnis im Gegensatz zu anderen Nutzern nicht unter `/home`, sondern unter `/opt/oracle`.

Damit die Installation und die Administration von Oracle durchgeführt werden kann, müssen in der `.profile` des Users `oracle` noch Pfadangaben für die Arbeit mit Oracle eingefügt werden. Da die `.profile` unter `/opt/oracle` noch nicht vorhanden ist, kann sie von einem anderen Nutzer in das Verzeichnis `/opt/oracle` kopiert werden, um die Umgebungsvariablen zu setzen. Diese Variablen überschreiben beim Login als oracle die unter `/etc/profile.d/oracle.sh` eingelesenen Variablen.

Kernelparameter, C-Bibliotheken und C-Compiler

Oracle benötigt unter Suse mindestens den Kernel 2.4.7 und glibc 2.2.2. Welcher Kernel vorhanden ist, erfährt man mit diesem Befehl:

```
uname -a
```

Die Version der auf Ihrem System vorhandenen C-Bibliotheken erfährt man über diesen Befehl:

```
/lib/libc.so.6
```

Auf unserem System war als C-Bibliothek die Version 2.2.4 installiert und der Kernel lief in der Version 2.1.10. Ein Update dieser Komponenten war also nicht nötig. Sollte dieses Update nötig sein, kann es über die Update Funktion von Yast/Yast2 unkompliziert durchgeführt werden. Anschließend wurde noch der vorhandene C-Compiler getestet und der Pfad zum Compiler ermittelt.

```
which gcc
```

Nun wird die Version des Gnu-C Compilers (gcc) ermittelt.

```
gcc -v -dumpversion
```

An dieser Stelle kommt jetzt auch das von Suse runtergeladene `rpm` zum Einsatz. Die Installation des Paketes erfolgen über diesen Befehl:

```
rpm -Uvh orarun.rpm
```

Erledigt wird dabei dieses:

- Die Umgebungsvariablen wie `$ORACLE_HOME` werden für jedes Login durch einen Eintrag in der Datei `/etc/profile.d/oracle.sh` gesetzt. Um jedoch diese Variablen exakt auf unser System und unsere Installation abzustimmen, haben wir bereits unter `/opt/oracle` die Datei `.profile` angepasst, die beim Login des Users oracle die Einträge in der `oracle.sh` überschreibt.
- Der entscheidende Punkt ist das setzen der Kernelparameter, wie z.B. max. shared memory (SHMMAX) auf einen Wert wie von Oracle gefordert. Dies kann auch manuell unter `root` erfolgen:
  `echo 3294967296 >/proc/sys/kernel/shmmax`
- Der Start und Stop der Oracle Datenbank soll bei startup und shutdown von Linux aufgerufen werden. Weitere Infos sind unter `/usr/share/doc/packages/orarun/README` zu finden.

Java Laufzeit Umgebung

In einigen Installation-Guides findet man den Hinweis, dass es zu Problemen kommt, wenn auf einem Linux System bereits ein `jre` installiert ist, da Oracle sein eigenes Runtime Environment mitbringt. Diese Probleme ließen sich unter Oracle 9.0.2 nicht reproduzieren und wurden offensichtlich in der aktuellen Oracle Distribution gefixt.

Entpacken der Dateien

Die vorhandenen drei Pakete müssen zunächst entpackt werden.

```
gunzip lnx_920_disk1.cpio.gz lnx_920_disk2.cpio.gz lnx_920_disk3.cpio.gz
```

Zusätzlich zur Komprimierung sind die benötigten Dateien nochmal in ein cpio Archiv zusammengefaßt worden. Nachdem die Dateien entpackt wurden, muß deshalb die im Archiv zusammengefaßte Verzeichnisstruktur über eine 'Dearchivierung' wieder hergestellt werden.

```
cpio -idmv < lnx_920_disk1.cpio
cpio -idmv < lnx_920_disk2.cpio
cpio -idmv < lnx_920_disk3.cpio
```

Danach sind im Installationsverzeichnis die Ordner Disk1, Disk2 und Disk3 sichtbar. Ab hier erfolgt jetzz die eigentlichie Installation.

runInstaller

Einloggen als User `oracle` mit X-Window. Um Probleme mit der Variable LANG zu vermeiden, ist diese zu löschen. Um die eigentliche Installation zu starten, ist ein Wechsel in das Verzeichnis Disk1 notwendig. Über `./runInstaller` wird die grafische Installation gestartet und das erste Fenster zur Benutzerführung erscheint.

```
umask 022
unset LANG
cd Disk1
./runInstaller
```

Oracle Universal Installer
Dialogbox 'Welcome'.

Dialogbox 'File Locations'
Source: ... Disk1/stage/products.jar
Destination (ORACLE_HOME): /opt/oracle/OraHome1
Hier wird ein Hauptverzeichnis vorgeschlagen. Dieser Vorschlag kann über einen klick auf OK angenommen werden.

Dialogbox 'Unix Group Name', Group: oinstall.
Eingabe des Gruppennamens, zu der der User oracle im Linux System gehört. In diesem Fall ist, da der User bereits angelegt ist und zur Gruppe oinstall und dba gehört, der Gruppenname oinstall. Der Schritt kann so akzeptiert werden.

Dateiverzeichnisse.
Angabe des Verzeichnisses, das als Installationsquelle dient und Angabe des Zielverzeichnisses.

Auswahl der Installationsprodukte.
Hier sollte für eine Installation des Oracle Servers und aller Management-Tools die oberste Option gewählt werden.

Auswahl der Installationsart.
Auch hier sollte die oberste Option (Enterprise Edition) ausgewählt werden.

Auswahl der Datenbank, die bei der Installation erstellt wird.
Es sollte 'General Purpose' ausgewählt werden, da hier eine vorkonfigurierte Datenbank für allgemeine Anwendungen installiert wird.

Dialogbox 'Choose JDK Home Directory'
In einer Konsole ermitteln, wo das JDK-Home-Directory ist:
`set | grep JDK_HOME`
Ggf. anpassen. Bei SuSE 8.2 ist es: `/usr/lib/java`

Dialogbox 'Database Identification'
Angabe des globalen Datenbanknamens im ersten Feld und eines Namens für die Datenbankinstanz im zweiten Feld. Creation Database
global database name: testdb
SID: testdb

Auswahl des Zeichensatzes.
Die oberste Option kann ausgewählt bleiben, da hier der Standard-Zeichensatz des Rechners ausgewählt ist.

Dialogbox 'Summary'
'Install' klicken.

Kurz vor Abschluß der Installation müssen einige Dateien und Verzeichnisse überschrieben werden. Deshalb muß ein Sript als user rrootäusgeführt werden, da dem User öracle die Berechtigung dafür fehlt. Das Script zur Ausführung heißt `root.sh` und ist unter `/opt/oracle/product/9.0.2` zu finden. Um es auszuführen muß die Konsole geöffnet werden und ein `su` auf `root` durchgeführt werden. Dann wird über `./root.sh` das Script ausgeführt. Die Abfrage, ob die Dateien, bzw. Verzeichnisse überschrieben werden sollen, mit Enter bestätigen und danach im noch offenen Fenster auf ok klicken, damit die Installation forgesetzt wird.

`/opt/oracle/OraHome1/root.sh` **hat aber einen Bug**:

```
RUID=‘/usr/bin/id|\$AWK -F\( ’{print \$2}’|\$AWK -F\) ’{print \$1}‘
```

Es fehlt ein Hochkomma vor dem letzten zurückgerichteten Hochkomma:

```
RUID=‘/usr/bin/id|\$AWK -F\( ’{print \$2}’|\$AWK -F\) ’{print \$1}’‘
```

Es werden noch einige Konfigurationswerkzeuge installiert, deren Installation optional ist. Die Installation dieser Werkzeuge kann auch später wiederholt werden.

Repository einrichten
`/opt/oracle/OraHome1/bin/emca`

Oracle Enterprise Manager - Configuration Assistant
...

Erstellen einer Datenbank

Um eine Datenbank unter Oracle zu erstellen, ist es sinnvoll, diese Einrichtung mit dem Datenbankassistenten vorzunehmen. Da die einzelnen Verzeichnisse bei der Einrichtung des Betriebssystems in den Pfad aufgenommen wurde, können wir jetzt von jeder beliebigen Stelle als user `oracle` einfach `dbca` in die Kommandozeile eintippen und der Datenbankkonfigurationsassistent wird gestartet.

```
dbca
```

An dieser Stelle kann einfach auf weiter geklickt werden.

Es werden unterschiedliche Optionen für die Verwaltung von vorhandenen Datenbanken angeboten. Da bisher noch keine Datenbank vorhanden ist, sind die Optionen 'Konfigurieren' und 'Löschen' ausgeblendet. Da eine Datenbank erstellt werden soll, wählen wir 'Datenbank erstellen'.

Dies ist ein wichtiger Schritt, denn ab Oracle 9i wird die Möglichkeit geboten Vorlagen zu erstellen, wie eine Datenbank beschaffen sein soll. Wir wählen hier die erste Option, da diese Vorlage zur allgemeinen Verwendung gedacht ist. Außerdem werden bei Auswahl dieser Option auch schon alle Datendateien erstellt.

Beim Klick auf 'Details' geht eine neues Fenster auf. Dieses Fenster enhält eine Auflistung aller Parameter für die Datenbankvorlage und kann bei Bedarf als html gespeichert werden.

Hier muß der Name für die Datenbank angegeben werden. Oracle hängt an diesen Namen später den Domainnamen ran, so das dieser Datenbankname später `meinname.meinedomain` lautet. Außerdem wird hier der Name für die Datenbankinstanz angegeben.
WICHTIG: Über diesen Namen, die sogenannte SID kann die Datenbank später referenziert werden. Das heißt, der Zugriff auf die Datenbank, z.B. von einem anderen Rechner aus, ist später über diese SID möglich.

Auswahl, in welchem Modus auf die Datenbank zugegriffen werden soll. Hier gibt es zwei Möglichkeiten:
Der Zugriff auf die Datenbank erfolgt über den **dedizierten Server Modus**. Das heißt, für jeden Client wird eine explizite Verbindung zur Datenbank hergestellt. Diese Einstellung ist sinnvoll, wenn nur wenige Clients gleichzeitig auf die Datenbank zugreifen.
Der Zugriff auf die Datenbank erfolgt über den sogenannten **Shared Server Modus**. (siehe Seite 158). Dieser Modus stellt eine Verbindungsverwaltung dar, die analog zu einem Connection Pooling funktioniert. Das bedeutet, eine Anzahl von Verbindungen wird zur Datenbank aufgebaut und ist immer vorhanden. Einem User wird dann eine Verbindung aus diesem Pool zugewiesen. Wird diese Verbindung vom User freigegeben, bleibt die Verbindung bestehen und wird an den nächsten User weitergegeben. Das hat den großen Vorteil, dass nicht immer wieder neue Verbindungen zur Datenbank aufgebaut und wieder freigegeben werden müssen. Option 1 wurde in diesem Fall von uns gewählt, da sie für eine Testinstallation völlig ausreicht.

An dieser Stelle werden die Initialisierungsparameter für die Datenbank angegeben. Dazu gehören die Speicherverwaltung, die Zeichensätze, die verwandt werden sollen, die DB-Skalierung (d.h. der maximale Speicherplatz, der für Sortiervorgänge genutzt werden kann, wobei höhere Werte die Effizienz von umfangreichen Sortiervorgängen erhöhen), die Dateispeicherorte und die Aktivierung des Archivelogmodus. Dieser Modus speichert Redo-Log Dateien, bevor

sie wiederverwandt werden. Im Notfall (Plattencrash, Verlust der Datenbank), kann so die Datenbank komplett wiederhergestellt werden. Dieser Modus ist automatisch aktiviert.

Einstellung der Parameter für die Speicherung der Datenbank. In einer Baumstruktur erhält man hier eine zusammenfassende Ansicht in der folgende Objekte angezeigt und angepaßt werden können:

- Kontrolldateien
- Tablespace
- Datendateien
- Rollback-Segmente
- Redo-Log-Gruppen

Dies ist der letzte Schritt vor der Erstellung der Datenbank. Standardmäßig ist die obere Checkbox mit einem Haken versehen und bei einem Klick auf die untere Checkbox gibt es die Möglichkeit eine eigene Datenbankvorlage zu speichern. Das heißt, es wird so etwas wie ein Template angelegt, dass später mit den eingestellten Parametern erneut aufgerufen werden kann und eine Datenbank erstellt, die den gespeicherten Vorgaben entspricht.

Beim Klick auf Beenden wird die Datenbank erstellt.

Nun müssen noch die Passwörter angegeben werden. Diese Passwörter gelten für den Benutzer SYS und für den Benutzer SYSTEM. Zur Sicherheit müssen die Passwörter wiederholt werden.
SYS Passwort: change_on_install
SYSTEM Passwort: manager

Nach dem Klick auf 'Beenden' ist die Erstellung der Datenbank abgeschlossen und die Passwörter sind gespeichert.

Konfigurieren von Oracle

Um überhaupt auf die Datenbank zugreifen zu können, ist es jetzt noch notwendig, eine Listenerprozeß einzurichten. Dafür steht ein Tool zur Verfügung, das über den Aufruf `netmgr` gestartet wird. Es erscheint jetzt ein Fenster das annähernd analog zum 'Net 8 Assistant' aufgebaut ist.

Wenn im rechten Fenster auf den Ordner 'Listener' geklickt wird und danach auf das grünze Kreuz auf der linken Seite, kann ein neuer Listenerprozeß eingerichtet werden. Dann im rechten Teil des Fensters 'Database Services' aus dem Dropdown Menü auswählen und bei den Registerkarten auf 'Add Database' klicken. Die bei der Installation der Datenbank angegebene SID kann jetzt im Feld SID eingetragen werden. Die Felder Global Database Name und Oracle Home Directory können so belassen werden. Ein Service Name für die Kennzeichnung der Verbindung zur Datenbank (unter Service Naming auf der linken Seite) muß nicht mehr angegeben werden. Im Gegensatz zu Oracle 8 wird unter der 9i bereits ein Servicename für die Datenbankinstanz angelegt, so dass unter diesem Namen auf die Datenbank zugegriffen werden kann, wenn ein Listenerprozeß angelegt wurde.

Datenbankinstanz und Listener starten

Zunächst muß in der Datei `/etc/oratab` folgende Zeile verändert werden:
`meindatenbankname:/opt/oracle/product/9.0.2:Y` (von N auf Y ändern)

Danach wird der Befehl dbstart ausgeführt.

```
dbstart
```

Es kommen einige Meldungen die den Status des Starts ausgeben. Die Meldungen schließen mit Database 'meindatenbankname' warm started Geschlossen wird die Datenbank über den Befehl `dbshut`.

```
dbshut
```

Wenn mehrere Datenbanken vorhanden sind, muß über den Befehl `export ORACLE_SID=sid` bekannt gemacht werden, welche Datenbank gestartet werden soll.

Der Listenerprozeß wird folgendermassen gestartet.

```
lsnrctl start LISTENER
```

Analog dazu wird er beendet.

```
lsnrctl stop LISTENER
```

Auch hier werden wieder Statusmeldungen ausgegeben. War der Start des Listeners erfolgreich, schließen diese Statusmeldungen mit dem Satz "Der Befehl wurde erfolgreich ausgeführt". Über den Befehl `ps ax | grep LISTENER` kann ich mir z.B. die PID des Listenerprozesses ausgeben lassen.

```
ps ax | grep LISTENER
```

Beim Aufruf des Befehls sqlplus kann ich jetzt über die Angabe von Benutzername/Passwort@Dienstname auf die Datenbank zugreifen.

```
sqlplus
....
sqlplus> .....: Benutzername/Passwort@Dienstname
```

1.0.2 Oracle-Tools

Die Oracle-Tools liegen im Path `$ORA_HOME/bin/*`

iSQL*Plus
`http://localhost:7778/isqlplus`
Siehe [8].

SQL*Plus-Worksheet
`oemapp worksheet`

SQL-Plus
`sqlplus user/passwort@db ... @*.sql`
`sqlplus ‘‘/sysdba`
Um mit den Beispieltabellen zu arbeiten, müssen Sie sich wie folgt einloggen:
`sqlplus sys@testd as sysdba`

oerr – Anzeige von Beschreibungen zu Oracle-Fehlermeldungen. Z.B. für den Fehler ORA-02291:

```
oerr ora 02291
02291, 00000,"integrity constraint (%s.%s) violated - parent key not found"
// *Cause: A foreign key value has no matching primary key value.
// *Action: Delete the foreign key or add a matching primary key.
```

Oracle Launch Pad
`olp`

Instance-Manager
`oemapp instance`

Server-Manager
`svrmgrl`

DBA-Studio
`oemapp dbastudio`

Console
`oemapp console`

Oracle Enterprise Manager – Configuration Assistant (z.B. Repository einrichten)
`emca`

netass
`netass`

netca
`netca`
Port einrichten und starten.

1.0.3 Datenbank-Design

Der Normalisierungsprozess

Die Normalisierung bezweckt die redundanzfreie Speicherung von Informationen innerhalb der Tabellen. Unter redundanzfreier Datenspeicherung versteht man, daß gleiche Informationen nicht mehrfach gespeichert werden.

1. Normalform
Eine Tabelle ist nach der ersten Normalform ausgerichtet, wenn alle elementarenInformationen in einzelne Felder aufgeteilt werden.

2. Normalform
Die zweite Normalform basiert auf der ersten Normalform und fordert eine eindeutige Identifikationsmöglichkeit der einzelnen Datensätze. Es dürfen keine Dateninhaltswiederholungen vorkommen.

3. Normalform
Die dritte Normalform basiert auf der zweiten Normalform und stellt eine weitere Verfeinerung dar. Es dürfen innderhalb einer Tabelle die Feldwerte nur vom Identifikationsschlüssel abhängig sein und untereinander keine Abhängigkeiten haben.

Beziehungen

In der Regel besteht eine relationale Datenbank nicht nur aus einer Tabelle, sondern aus mehreren. Die einzelnel Tabellen dürfen nicht isoliert betrachtet werden. Zwischen Tabellen können Beziehungen bestehen. Die Anzahl der möglichen Beziehungen ist begrenzt und ergibt sich aus der Kombination der möglichen Assioziationstypen.

Assoziation
Eine Assoziation bestimmt, wieviel Datensätze einer Tabelle 2 zu einem Datensatz der Tabelle 1 gehören.

1 – einfache Assoziation
Datensatzanzahl in Tabelle 2 – genau ein Datensatz (1)

c – konditionelle Assoziation
Datensatzanzahl in Tabelle 2 – kein oder genau ein Datensatz (0/1)

m – multiple Assoziation
Datensatzanzahl in Tabelle 2 – mindestens ein Datensatz (>= 1)

mc – multiple-konditionelle Assoziation
Datensatzanzahl in Tabelle 2 – beliebig viele Datensätze (>= 0)

1.0.4 Übersicht über die Beispieltabellen

Tabellenstrukturen der DB Scott

```
EMP
-----------------------------------------------
EMPNO          NUMBER(4) NOT NULL
ENAME          VARCHAR2(10)
JOB            VARCHAR2(9)
MGR            NUMBER(4)
HIREDATE       DATE
SAL            NUMBER(7,2)
COMM           NUMBER(7,2)
DEPTNO         NUMBER(2)

DEPT
-----------------------------------------------
DEPTNO         NUMBER(2) NOT NULL
DNAME          VARCHAR2(14)
LOC            VARCHAR2(13)

SALGRADE
-----------------------------------------------
GRADE NUMBER
LOSAL NUMBER
HISAL NUMBER
```

Tabellenstrukturen der DB Nordwind

```
customers
-----------------------------------------------
CustomerID                varchar2(5)  NOT NULL
CompanyName               varchar2(40) NOT NULL
ContactName               varchar2(30) NULL
ContactTitle              varchar2(30) NULL
Address                   varchar2(60) NULL
City                      varchar2(15) NULL
Region                    varchar2(15) NULL
PostalCode                varchar2(10) NULL
Country                   varchar2(15) NULL
Phone                     varchar2(24) NULL
Fax                       varchar2(24) NULL

orders
-----------------------------------------------
OrderID                   int          NOT NULL
CustomerID                varchar2(5)  NULL
EmployeeID                int          NULL
OrderDate                 date         NULL
RequiredDate              date         NULL
ShippedDate               date         NULL
ShipVia                   int          NULL
Freight                   number(20,2) NULL
ShipName                  varchar2(40) NULL
ShipAddress               varchar2(60) NULL
```

```
ShipCity                  varchar2(15) NULL
ShipRegion                varchar2(15) NULL
ShipPostalCode            varchar2(10) NULL
ShipCountry               varchar2(15) NULL


order_details
-------------------------------------------------
OrderID                   int          NOT NULL
ProductID                 int          NOT NULL
UnitPrice                 number(20,2) NOT NULL
Quantity                  int          NOT NULL
Discount                  number(20,5) NOT NULL


products
-------------------------------------------------
ProductID                 int          NOT NULL
ProductName               varchar2(40) NOT NULL
SupplierID                int          NULL
CategoryID                int          NULL
QuantityPerUnit           varchar2(20) NULL
UnitPrice                 number(20,2) NULL
UnitsInStock              int          NULL
UnitsOnOrder              int          NULL
ReorderLevel              int          NULL
Discontinued              int          NOT NULL


categories
-------------------------------------------------
CategoryID                int          NOT NULL
CategoryName              varchar2(15) NOT NULL
Description               varchar2(255)


suppliers
-------------------------------------------------
SupplierID               int   NOT NULL
CompanyName              varchar2(40)  NOT NULL
ContactName              varchar2(30)  NULL
ContactTitle             varchar2(30)  NULL
Address                  varchar2(60)  NULL
City                     varchar2(15)  NULL
Region                   varchar2(15)  NULL
PostalCode               varchar2(10)  NULL
Country                  varchar2(15)  NULL
Phone                    varchar2(24)  NULL
Fax                      varchar2(24)  NULL
HomePage                 varchar2(255) NULL


shippers
-------------------------------------------------
ShipperID                int          NOT NULL
CompanyName              varchar2(40)  NOT NULL
Phone                    varchar2(24)  NULL


employees
-------------------------------------------------
```

```
EmployeeID              int             NOT NULL
LastName                varchar2(20)    NOT NULL
FirstName               varchar2(10)    NOT NULL
Title                   varchar2(30)    NULL
TitleOfCourtesy         varchar2(25)    NULL
BirthDate               date            NULL
HireDate                date            NULL
Address                 varchar2(60)    NULL
City                    varchar2(15)    NULL
Region                  varchar2(15)    NULL
PostalCode              varchar2(10)    NULL
Country                 varchar2(15)    NULL
HomePhone               varchar2(24)    NULL
Extension               varchar2(4)     NULL
ReportsTo               int             NULL

employeeTerritories
-------------------------------------------------
EmployeeID              int             NOT NULL
TerritoryID             varchar2(20)    NOT NULL

territories
-------------------------------------------------
TerritoryID             varchar2(20)    NOT NULL
TerritoryDescription    varchar2(255)   NOT NULL
RegionID                int             NOT NULL

region
-------------------------------------------------
RegionID                int             NOT NULL
RegionDescription       varchar2(50)    NOT NULL
```

Die Scripte zum Erzeugen dieser Datenbank-Tabellen befinden sich in [3].

1.0.5 SQL Statements

Groß- und Kleinschreibung sind bei den Anweisungen nicht relevant.

Data Retrieval

SELECT

Data Manipulation Language (DML)

INSERT

UPDATE

MERGE

DELETE

Data Definition Language (DML)

CREATE

DROP

ALTER

RENAME

TRUNCATE

Transaction Control

COMMIT

ROLLBACK

SAVEPOINT

Data Control Language (DCL)

GRANT

REVOKE

1.1 Data Retrieval

Einfache SQL-Abfragen.

1.1.1 Kommentare

```
--
/*
*/
```

1.1.2 SELECT FROM – Spalten einer Tabelle selektieren

Eine der Hauptaufgaben in einer Datenbank ist das Abfragen der gewünschten Datensätze. Hierfür stellt SQL die SELECT ... FROM Anweisung zur Verfügung. Eine SELECT-Anweisung gibt als Ergebnismenge eine, mehrere oder alle Spalten einer Tabelle zurück (Auswahl).

Zwischen den beiden Schlüsselworten SELECT und FROM stehen die Spalten, die zurückgegeben werden sollen. Hier können auch mathematische Ausdrücke und Literale angegeben werden.

```
select spaltenname1, spaltenname2 from tabellenname;
```

Sollen alle Spalten selektiert werden, so setzen Sie anstelle der Spaltenname einfach den Stern *.

```
select * from dept;
```

Es werden alle Spalten der Tabelle dept angezeigt.

```
select deptno, dname from dept;
```

Es werden die Abteilungsnummern (deptno) und Abteilungsnamen (dname) aller Abteilungen angezeigt. Sollten sich in Aliasnamen Sonderzeichen oder Leerzeichen befinden, so ist der Aliasname in Anführungzeichen zu setzen.

Die Überschriften der Spalten können durch einen Alias angegeben werden (spaltenname AS aliasname). Das Wort AS kann bei Aliasnamen weggelassen werden.

```
select deptno as "Abteilungsnummer", dname as "Abteilungsname" from dept;
```

Es werden die Abteilungsnummern (deptno) und Abteilungsnamen (dname) aller Abteilungen angezeigt. Die Spaltenüberschrift von depto ist Abteilungsnummer und die Spaltenüberschrift von dname ist Abteilungsname.

Zwischen SELECT und FROM kann auch eine Zeichenkette stehen. Diese muß in einfache Anführungszeichen gesetzt werden. Zwischen SELECT und FROM kann auch ein arithmetischer Ausdruck stehen.

```
select empno as "Mitarbeiternummer",
       ename as "Mitarbeitername",
       sal as "Gehalt",
       sal*1.3 as "Bruttogehalt"
   from emp
;
```

Es werden die Spalten Mitarbeiternummer (empno), Mitarbeiternamen (ename), Gehalt (sal) und Gehalt*1.3 angezeigt. Mehrfache Zeilen werden durch DISTINCT direkt nach dem SELECT ausgeblendet.

```
select distinct job from emp;
```

Es werden alle Jobs (job) der Tabelle emp angezeigt. Da einige Mitarbeiter den gleichen Job haben, tauchen diese Jobs dementsprechend mehrfach auf. Die Duplikate werden durch DISTINCT nicht angezeigt. Eine Verkettung von Spalten ist mit || möglich.

```
select ename || ' ist ein ' || job from emp;
```

Es wird die Spalte ename, verkettet mit dem Literal ' ist ein' und mit job, angezeigt.

1.1.3 DESCRIBE – Tabellenstruktur anzeigen lassen

Häufig ist es wichtig, einen Überblick über den Aufbau einer Tabelle zu verschaffen. DESCRIBE beschreibt die Struktur der entsprechenden Tabelle und gibt Auskunft über Spaltennamen, Datentypen und NULL-Werten.

```
describe tabellenname;
```

```
describe employees;

Name    Null?      Typ
------------------------------
EMPNO   NOT NULL   NUMBER(4)
ENAME              VARCHAR2(10)
JOB                VARCHAR2(9)
MGR                NUMBER(4)
HIRE               DATEDATE
SAL                NUMBER(7,2)
COMM               NUMBER(7,2)
DEPTNO             NUMBER(2)
```

1.1.4 WHERE – Zeilen der Ergebnismenge einschränken

Mit Hilfe des Schlüsselwortes WHERE ist es möglich, die Datensätze, die zurückgegeben werden sollen, einzuschränken. Hinter WHERE wird ein Ausdruck eingegeben, der jeweils ausgewertet wird und die Stati Wahr (true) oder falsch (false) ausgibt. Die Datensätze, bei denen der Ausdruck wahr ergibt, werden als Ergebnismenge zurückgegeben. Eine Bedingung muss ein gültiger Ausdruck sein (z.B. job='MANAGER'). In diesem Ausdruck können folgende Vergleichsoperatoren benutzt werden (= gleich, >größer als, >= größer oder gleich, <kleiner, <= kleiner oder gleich, <>ungleich).

```
select spaltenname1, spaltenname2 from tabellenname where bedingung;
```

```
select ename, job from emp where job = 'MANAGER';
```

Ermittelt werden alle Mitarbeiter (ename), bei denen der Job Manager ist.

```
select * from emp where hiredate = '03.12.81';
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen das Einstellungsdatum (Hiredate) 3.12.81 ist.

```
select * from emp where sal >= 1100;
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen das Gehalt größer bzw. gleich 1100 ist.

```
select ename, job from emp where job <> 'MANAGER';
```

Ermittelt werden alle Mitarbeiter (ename), bei denen der Job nicht Manager ist.

BETWEEN

Der Vergleichsoperator BETWEEN ermittelt alle Datensätze zwischen zwei Vergleichswerten, einschließlich der Randwerte.

```
select * from emp where sal between 1000 and 2000;
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen das Gehalt >= 1000 und <= 2000 ist.

IS NULL

Mit IS NULL kann man Datenfelder auf den Wert NULL abtesten.
Mit "... = NULL"wäre dies nicht möglich, da hier auf das Vorkommen der Zeichenkette NULL getestet wird.

```
select empno, ename from emp where comm is null;
```

Ermittelt werden alle Mitarbeiter (ename) und deren Mitarbeiternummer (empno), bei denen die Spalte comm den Wert NULL besitzt.

IN

Der Vergleichsoperator IN vergleicht mit einer Liste.

```
select * from employees where city in ('London','Berlin');
```

Es werden alle Datensätze angezeigt, bei denen die Spalte city gleich London oder Berlin ist.

Übungen 01

Übungen siehe Seite 238.

1.1.5 LIKE – Vergleich mit Platzhaltern % _

Wenn Sie in der Unix-Shell nicht genau wissen, wonach Sie suchen wollen, benutzen Sie in der Regel Platzhalter. Als Platzhalter für ein Zeichen benutzen Sie dabei das Fragezeichen ? und als Platzhalter für eine beliebige Zeichenkette benutzen Sie den Stern *. In Oracle-SQL verhält sich dies ein wenig anders. Als Platzhalter für ein Zeichen gilt hier der Unterstrich _ und als Platzhalter für eine beliebige Zeichenkette dient das Prozentzeichen %.

```
select * from emp where ename like 'F%'
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen der Name mit 'F' beginnt.

```
select * from emp where ename like 'F_rd'
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen der Name mit 'F' beginnt und an dritter und vierter Stelle 'rd' steht.

```
select * from emp where ename like '%g'
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen der Name mit 'g' endet.

ESCAPE

Durch ESCAPE kann man nach _ und % suchen und vergleichen.

```
select * from emp where ename like '%/_%' ESCAPE '/'
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen ein Unterstrich _ enthalten ist.

Übungen 02

Übungen siehe Seite 238.

1.1.6 AND, OR, NOT – Logische Operatoren

SQL ist eine mengenorientierte Abfragesprache. Daher baut diese Sprache auch auf Boolsche Logik auf. Der Ausdruck im WHERE-Kriterium kann logische Operatoren, wie AND, OR und NOT beinhalten. Beachten sollten Sie, dass der Ausdruck hinter WHERE wahr ergeben muss, um den Datensatz zurückgeben zu können. Bei AND müssen beide Ausdrücke wahr sein, um als Ergebnis wahr zurückzuliefern. Bei OR muss lediglich ein Ausdruck wahr sein, um als Ergebnis wahr zurückzuliefern. NOT negiert das Ergebnis, das heißt, aus Wahr wird falsch und aus falsch wird wahr.

```
select * from emp where sal > 1000 and sal < 3000;
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen das Gehalt größer als 1000 und kleiner als 3000 ist.

```
select * from emp where sal > 1000 and deptno = 30;
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen das Gehalt größer als 1000 und die Abteilungsnummer (deptno) 30 ist.

```
select * from emp where sal < 1000 or sal > 3000;
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen das Gehalt kleiner als 1000 oder größer als 3000 ist.

```
select * from emp where not sal > 1000;
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen das Gehalt nicht größer als 1000 ist.

```
select * from emp where job not in ('clerk','manager');
```

Ermittelt werden alle Datensätze der Tabelle emp, bei denen der Job weder CLERK noch MANAGER ist.

Prioritätsregeln

Wie schon aus der Schulmathematik bekannt, geht Punktrechnung vor Strichrechnung. Bei SQL kommen nun neben dieser Regel auch noch die Logikoperatoren ins Spiel. Die Reihenfolge der Auswertung ist:

1. Vergleichsoperatoren (=, <>, >=, <=).
2. NOT
3. AND
4. OR

Die Reihenfolge der Anordnung spielt hierbei keine Rolle. Das heißt, sal >1000 and sal <3000 and sal <>500 entspricht sal <>500 and sal >1000 and sal <3000.

```
select * from emp where sal >= 1000 and sal <= 3000 or sal > 5000;
```

Diese Anweisung ermittelt die Mitarbeiter, deren Gehalt zwischen 1000 und 3000 liegt oder deren Gehalt größer als 5000 ist. Durch Klammersetzung kann die Reihenfolge der Abarbeitung verändert werden.

```
select * from emp where sal >= 1000 and (sal <= 3000 or sal > 5000);
```

Diese Anweisung ermittelt die Mitarbeiter, deren Gehalt entweder kleiner gleich 3000 oder größer 5000 ist. Es werden nur die Datensätze angezeigt, deren Gehalt zusätzlich größer gleich 1000 ist.

1.1.7 ORDER BY – Sortierung in SQL

Die Datenbanktheorie besagt, dass die Reihenfolge der Datensätze, die zurückgegeben werden, 'zufällig' ist. Das bedeutet, dass sich das Datenbankmanagementsystem das Recht vorbehält, die Datensätze in einer beliebigen Reihenfolge zurückzugeben.

Wird eine bestimmte Reihenfolge bei der Rückgabe der Datensätze gewünscht, so ist dies entsprechend durch ein Schlüsselwort ORDER BY anzugeben. Es können mehrere ORDER BY Spalten angegeben werden. Auch ist es möglich, anstelle des Spaltennamens hier den numerischen Spaltenindex anzugeben. Es kann auch nach einem Alias sortiert werden.

ASC hinter dem Sortierkriterium heißt aufsteigend sortieren. Standardmäßig wird aufsteigend sortiert. DESC hinter dem Sortierkriterium heißt absteigend sortieren.

Bei alphanumerischen Zeichen wird von A bis Z an den entsprechenden Stellen sortiert. Das früheste Datum wird bei aufsteigender Sortierung als erstes angezeigt. NULL-Werte werden bei aufsteigender Sortierung als letzte angezeigt und bei absteigender Sortierung demnach als erste.

```
select spaltenname1, spaltenname2 from tabellenname
  order by spaltenname_x, spaltenname_y .. ;
```

```
select empno, ename from emp order by empno;
select empno, ename from emp order by empno asc;
```

Es werden die Mitarbeiter aufsteigend sortiert nach der Mitarbeiternummer (empno) angezeigt.

```
select empno, ename from emp order by empno desc;
```

Es werden die Mitarbeiter absteigend sortiert nach der Mitarbeiternummer (empno) angezeigt

```
select empno, ename from emp order by job,hiredate;
```

Es werden die Mitarbeiter aufsteigend sortiert nach ihrem Job (job) angezeigt. Bei Mitarbeitern mit gleichem Job wird nochmals aufsteigend nach dem Einstellungsdatum sortiert.

```
select empno as Mitarbeiternummer, ename from emp
  order by Mitarbeiternummer
;
```

Es werden die Mitarbeiter aufsteigend sortiert nach der Mitarbeiternummer (empno) angezeigt. Als Sortierkriterium wurde hier der Alias einer Spalte benutzt.

```
select empno as Mitarbeiternummer, ename from emp order by 2;
```

Es werden die Mitarbeiter aufsteigend sortiert nach ihrem Namen (ename) angezeigt. Als Sortierkriterium wurde hier der Spaltenindex benutzt (2 Spalte zwischen select und from).

1.2 Single Row Functions

Eine Übersicht befindet sich in [14].

1.2.1 Character / String Functions

Häufig ist es notwendig, die Spalten der Ergebnismenge gemäß einer Vorgabe zu gestalten. Hierfür stellt SQL eine Reihe von Funktionen zur Verfügung. Beachten Sie bitte, dass sich diese Funktionen nur auf die Anzeige auswirken, nicht jedoch auf die eigentlichen gespeicherten Daten. Diese werden in keiner Weise verändert.

Zeichenkettefunktionen können nicht nur zwischen SELECT und FROM benutzt werden, sondern auch im WHERE-Ausdruck. Im folgenden sehen wir uns drei Zeichenkettenfunktionen bezugnehmend auf Groß- und Kleinschreibung an.

LOWER()

Mit der Funktion LOWER wird eine Zeichenkette in Kleinbuchstaben umgewandelt.

```
select lower(ename) as Nachname from emp;
NACHNAME
--------
smith
allen
ward
...
```

Diese Abfrage gibt als Ergebnismenge die Nachnamen (ename) der Mitarbeiter in Kleinbuchstaben zurück

UPPER()

Es wird durch die Funktion UPPER eine Zeichenkette in Großbuchstaben umgewandelt.

```
select upper(ename) as Nachname from emp;
NACHNAME
--------
SMITH
ALLEN
WARD
...
```

Diese Abfrage gibt als Ergebnismenge die Nachnamen (ename) der Mitarbeiter in Großbuchstaben zurück.

INITCAP()

INITCAP gibt den ersten Buchstaben groß, alle weiteren klein geschrieben zurück.

```
select initcap(ename) as Nachname from emp;
NACHNAME
--------
Smith
Allen
Ward
...
```

Diese Abfrage gibt als Ergebnismenge die Nachnamen (ename) der Mitarbeiter zurück. Der erste Buchstabe wird groß geschrieben und alle weiteren werden klein geschrieben.

Zusammenfassung

- LOWER() wandelt die Anzeige der Spalte / des Ausdruckes x in Kleinbuchstaben um.
- UPPER() wandelt die Anzeige der Spalte / des Ausdruckes x in Großbuchstaben um.
- INITCAP() wandelt das erste Zeichen der Anzeige der Spalte / des Ausdruckes x in Großbuchstaben um – alles andere bleibt klein.

Neben den im vorigen Kapitel besprochenen Zeichenkettenfunktionen gibt es noch eine große Anzahl weiterer Funktionen. Hier soll nur ein Auszug der wichtigsten Funktionen dargestellt werden.

CONCAT()

```
select concat(ename,job) as Nachname from emp;
NACHNAME
--------------
SMITHCLERK
ALLENSALESMAN
WARDSALESMAN
...
```

Es werden die Spalten Nachname (ename) und Job verknüpft. Dies entspricht dem Verknüpfungsoperator `||`.

SUBSTR()

```
select ename as Nachname, substr(ename,1,3) from emp;
NACHNAME   SUB
-----------------
SMITH      SMI
ALLEN      ALL
WARD       WAR
...
```

Diese Abfrage gibt eine Zeichenkette von 3 Zeichen, beginnend beim ersten Zeichen, zurück.

```
select ename as Nachname, substr(ename,3,2) from emp;
NACHNAME   SU
-------------
SMITH      IT
ALLEN      LE
WARD       RD
...
```

Diese Abfrage gibt eine Zeichenkette von 2 Zeichen, beginnend beim dritten Zeichen, zurück.

```
select ename as Nachname, substr(ename,-3,2) from emp;
NACHNAME   SU
-------------
SMITH      IT
ALLEN      LE
WARD       AR
...
```

Diese Abfrage gibt eine Zeichenkette von 2 Zeichen, beginnend beim dritten Zeichen von hinten, zurück.

INTSTR()

```
select ename as Nachname, instr(ename,'S') from emp;
NACHNAME INSTR(ENAME,'S')
--------------------------
SMITH       1
ALLEN       0
WARD        0
...
```

Es wird in der Spalte Nachname (ename) nach dem Vorkommen von 'S' gesucht und die Position zurückgegeben. Wird ‚S' nicht gefunden, so wird eine 0 zurückgegeben.

LENGHT()

```
select ename as Nachname, length(ename) as Länge from emp;
NACHNAME  LÄNGE
---------------
SMITH       5
ALLEN       5
WARD        4
...
```

Es wird die Länge der Spalte Nachname (ename) zurückgegeben.

```
SELECT productname, length(productname) AS Laenge FROM products;
PRODUCTNAME  LAENGE
------------------
Milch            20
Käse             20
...
```

Es wird die Länge der Daten in der Spalte productname ermittelt. Da der Datentyp der Spalte productname aber CHAR ist, wird der verbleibende Platz bis zur bei CHAR angegebenen Länge mit Leerzeichen aufgefüllt. LENGTH gibt bei CHAR also immer die gleiche Länge aus, nämlich den Wert, der bei CHAR angegeben wurde.

TRIM()

```
select trim(' ' from '   Tech    ') from dual;
```

Es werden die vorangestellten und nachfolgenden Leerzeichen entfernt.

```
SELECT productname,length(trim (' ' from productname) AS Laenge
 FROM products;
PRODUCTNAME  LAENGE
-----------------------
Milch             5
Käse              4
...
```

Es wird die Länge der Daten in der Spalte productname ermittelt. TRIM schneidet diese Leerzeichen ab. Es wird demnach als Ergebnis die echte Länge angegeben.

LTRIM()

```
select ltrim('    Tech') from dual;
```

Es werden die vorangestellten Leerzeichen entfernt.

RTRIM()

```
select rtrim('Tech    ') from dual;
```

Es werden die angefügten Leerzeichen entfernt.

RPAD()

```
select rpad('Tech',10,'*') from dual;
```

Es werden bis zur Gesamtlänge von 10 Zeichen mit * rechts aufgefüllt.

LPAD()

```
select lpad('Tech',10,'*') from dual;
```

Es werden bis zur Gesamtlänge von 10 Zeichen mit * links aufgefüllt.

REPLACE()

```
replace(s1,s2[,s3])
```

Suche s2 in s1 und ersetze ihn durch s3. Ist s3 nicht angegeben, wird s2 in s1 entfernt.

```
select replace('SCHADE', 'D', 'LK') from dual;
'SCHALKE'.
```

TRANSLATE()

```
translate(s1,s2,s3)
```

In s1 werden alle Zeichen aus s2 durch solche aus s3 ersetzt.

```
select translate('ABC67LR5', '0123456789','**********') from dual;
'ABC**LR*'
```

```
select translate('ABC67LR5', '*0123456789','*') from dual;
'ABCLR'.
```

CHR()

```
select chr(65) from dual;
'A'
```

ASCII()

```
select ascii('A') from dual;
65
```

1.2.2 Conversion Functions

Wichtige Zahlenformate

Format	Beschreibung	Beispiel
999999	5 Stellen	1234
099999	Auffüllen mit Nullen	001234

Häufige Zahlenformate

9	Stellvertreter für eine Ziffer (keine führenden Nullen)
0	Stellvertreter für eine Ziffer (Führende Nullen)
D	Dezimaltrennzeichen
G	Tausenedertrennzeichen
L	Währung ??

Wichtige Datumsformate

DD	zweistelliger Tageswert
DAY	ausgeschriebener Tag
MM	zweistelliger Monatswert
Month	ausgeschriebener Monat
YY	zweistelliger Jahreswert
YYYY	vierstelliger Jahreswert

Beispiele für Datumsformate:
DD.MM.YYYY – 11.02.2003
Day, “der”DD.MM.YYYY – Dienstag, der 11.02.2003
Day, “der”DD Month YYYY – Dienstag, der 11 Februar 2003

TO_CHAR()

```
select ename as "Nachname", to_char(sal,'09999') as Gehalt from emp;
Nachname   GEHALT
-----------------
SMITH       00800
ALLEN       01600
...
```

Es wird das Gehalt der Angestellten in eine Zeichenkette konvertiert und mit 5 Stellen vor dem Komma (mit Tausender-Trennzeichnen) und zwei Stellen nach dem Dezimalzeichen ausgegeben.

```
select ename as "Nachname", to_char(sal,'09G999D00') as Gehalt from emp;
```

Es wird das Gehalt der Angestellten in eine Zeichenkette konvertiert und mit 5 Stellen angegeben. Falls das Gehalt nicht 5 Stellen beträgt, so wird es mit führenden Nullen aufgefüllt. Dies ist zum Beispiel bei der PLZ in Deutschland sinnvoll.

```
select ename as "Nachname",
       to_char(hiredate,'Day, "dem" dd. Month yyyy') as "Eingestellt am"
  from emp
;
Nachname  Eingestellt am
-------------------------------------------
SMITH     Mittwoch , dem 17. Dezember  1980
ALLEN     Freitag  , dem 20. Februar   1981
...
```

Es wird das Einstellungsdatum der Angestellten in eine Zeichenkette konvertiert und im Format 'Day, “dem“dd. Month yyyy' ausgegeben.

TO_NUMBER()

```
select to_number('   1,5') as Zahl from dual;
ZAHL
-----
 1,5
```

Die Zeichenkette ' 1,5' wird in eine Zahl konvertiert.

TO_DATE()

```
select to_date('01.01.2001','dd.mm.yyyy') as Datum from dual;
DATUM
--------
01.01.01
```

Die Zeichenkette '01.01.2001' wird gemäß des Formates 'dd.mm.yyyy' als Datum interpretiert und in ein Datum konvertiert.

1.2.3 Advanced Functions

NVL()

Mit Hilfe der Funktion **NVL** ist es möglich, NULL-Werte durch beliebige Werte zu ersetzen. NVL(Comm,0) ersetzt als Beispiel alle NULL-Werte in der Spalte Comm durch eine echte 0. Betont werden soll an dieser Stelle nochmals, dass das Ersetzen sich nicht auf die gespeicherten Datensätze auswirkt.

```
select nvl(comm,0) as "Kein NULL mehr" from emp;
Kein NULL mehr
--------------
     0
   300
   500
   ...
```

Es wird in der Spalte comm nach NULL gesucht und durch eine 0 bei der Ausgabe ersetzt.

NVL2()

Mit Hilfe der Funktion **NVL2** ist es möglich, abhängig vom Inhalt einer Spalte unterschiedliche Werte zurückzugeben. Hat die Spalte s1 einen Inhalt, so wird s2 zurückgegeben. Ist s1 NULL, so wird s3 zurückgegeben.

```
nvl2(s1,s2,s2)
```

```
select comm,nvl2(comm,'Komm','Keine Komm') from scott.emp;
```

DECODE()

Durch Decode werden verschiedene Ausdrücke in Abhängigkeit eines Kriteriums ausgewertet. Sie können sich die Funktion DECODE wie eine Art IF ... THEN ... ELSE vorstellen.

```
select ename as Nachname,
       job,
       sal as "Altes Gehalt",
       decode (job,'CLERK',sal*0.9,'SALESMAN',sal*0.8,sal) as "Gekürztes Gehalt"
  from emp
;
NACHNAME  JOB       Altes Gehalt Gekürztes Gehalt
------------------------------------------------
SMITH     CLERK           800            720
ALLEN     SALESMAN       1600           1280
WARD      SALESMAN       1250           1000
JONES     MANAGER        2975           2975
...
```

Es wird in der Spalte Job nach clerk gesucht und in diesen Zeilen das Gehalt mit 0.9 multipliziert. Weiterhin wird in der Spalte Job nach salesman gesucht und in diesen Zeilen das Gehalt mit 0.8 multipliziert. Für alle anderen Zeilen gilt, dass das Gehalt ungekürzt ausgegeben wird.

1.2.4 Mathematical Functions

SQL bietet eine Reihe arithmetischer Funktionen. Anhand der folgenden Beispiele soll das Runden **ROUND** und Abschneiden **TRUNC** genauer analysiert werden.

ROUND()

```
select 12345.6789 as Zahl, round(12345.6789,2) as Gerundet
  from dual
;
ZAHL         GERUNDET
---------------------
12345,6789   12345,68
```

Die Zahl 12345.6789 wird auf zwei Nachkommastellen gerundet.

```
select 12345.6789 as Zahl, round(12345.6789,-2) as Gerundet
  from dual
;
ZAHL        GERUNDET
--------------------
12345,6789     12300
```

Die Zahl 12345.6789 wird auf zwei Stellen vor dem Komma gerundet.

TRUNC()

```
select 12345.6789 as Zahl, trunc(12345.6789,2) as Abgeschnitten
  from dual
;
ZAHL      ABGESCHNITTEN
----------------------
12345,6789     12345,67
```

Die Zahl 12345.6789 wird auf zwei Nachkommastellen abgeschnitten.

```
select 12345.6789 as Zahl, trunc(12345.6789,-2) as Abgeschnitten
  from dual
;
ZAHL        ABGESCHNITTEN
------------------------
12345,6789          12300
```

Die Zahl 12345.6789 wird auf zwei Stellen vor dem Komma abgeschnitten.

1.2.5 Date Functions

Datumsberechnungen

Datumswerte werden in Oracle intern als numerische Werte interpretiert. Hierbei entspricht eine ganze Zahl einem vollen Tag. Die Zahl 0,5 würde als 12 Stunden interpretiert werden, die Zahl 0,75 würde als 18 Stunden interpretiert werden. Bei Berechnungen mit Datumswerten gelten folgende Eigenschaften:

```
Datum + Zahl  = Datum
Datum - Zahl  = Datum
Datum - Datum = Anzahl der Tage
```

```
select sysdate-1 as "Gestern", sysdate as "Heute", sysdate+1 as "Morgen"
  from dual
;
 Gestern     Heute    Morgen
---------------------------
26.12.03  27.12.03  28.12.03
```

Es werden die Dati von Gestern, Heute und Morgen ausgegeben.

```
select sysdate - hiredate as "Tage als Angestellter" from emp;
Tage als Angestellter
---------------------
8091,85933
8026,85933
...
```

Es wird vom aktuellen Datum (sysdate) das Einstellungsdatum abgezogen. Ausgegeben wird nun die Anzahl von Tagen zwischen dem aktuellen Datum (sysdate) und dem Einstellungsdatum.

MONTHS_BETWEEN()

Mit der Funktion MONTHS_BETWEEN ist es möglich, die Monate zwischen zwei Datumswerten zu ermitteln.

```
select ename as "Nachname",
       months_between(sysdate,hiredate)/12 as "Jahre als Angestellter"
  from emp
;
Nachname  Jahre als Angestellter
------------------------------
SMITH              22,1528973
ALLEN              21,9781661
```

Es werden die Monate zwischen dem aktuellen Datum (sysdate) und dem Einstellungsdatum ermittelt. Das Ergebnis sind demnach die Monate, die die Person Angestellt ist. Dieser Wert wird durch 12 geteilt, und als Ergebnis ist nicht mehr in Monaten, sondern in Jahren.

ADD_MONTHS()

Mit ADD_MONTHS können eine bestimmte Anzahl von Monaten auf ein Datum addiert werden.

```
select ename as "Nachname",add_months(hiredate,12*25) as "Gehaltserhöhung"
  from emp
;
Nachname  Gehaltserhöhung
-------------------------
SMITH         17.12.05
ALLEN         20.02.06
```

Zu dem Einstellungsdatum werden 12*25 Monate, also 25 Jahre, addiert. Dieses Datum wird ausgegeben. Es wird also der Tag 25 Jahre nach dem Einstellungsdatum ausgegeben. In diesem Fall ist an diesem Tag eine Gehaltserhöhung vom Chef geplant.

Übungen 06

Übungen siehe Seite 248.

1.3 Komplexere SQL-Abfragen

1.3.1 JOIN – Verbinden von Tabellen

Unter einem Join versteht man das mathematische Mittel der Projektion. Dies bedeutet, dass die Ergebnismenge sich aus Spalten verschiedener Tabellen zusammensetzt. Nehmen wir einmal an, es gebe die Tabellen Kunden, Bestellungen und Produkte. Nun wollen Sie wissen, welcher Kunden (Kundenname) welches Produkt (Produktname) bestellt hat. Da diese Informationen sich in diesem Fall über drei Tabellen verteilen, ist eine Abfrage über alle beteiligten Tabellen notwendig.

Arten

Es gibt zwei Hauptarten:

- EQUIJOIN
- NON-EQUIJOIN

Weiterhin gibt es noch JOIN-Methoden:

- Outer Join
- Self Join
- Set Operators

Set Operators ist nicht Bestandteil dieses Kurses. Beachte den Unterschied zwischen Outer Join und Full Outer Join (Kreuzprodukt).

EQUIJOIN über zwei Tabellen

Bei einem EQUIJOIN wird auf Gleichheit getestet. Anhand des Beispiels Mitarbeiter (emp) und Abteilung (dept) soll dieser Sachverhalt dargestellt werden. Als Ergebnis wird eine Menge gewünscht, in der in der ersten Spalte der Mitarbeitername (ename aus emp) und in der zweiten Spalte der Abteilungsname (dname aus dept) angezeigt wird. Natürlich soll immer der zu diesem Mitarbeiter gehörende korrekte Abteilungsname angezeigt werden. Schauen wir uns die beiden beteiligten Tabellen etwas genauer an

In der Tabelle emp gibt es eine Spalte deptno, die der Abteilungsnummer entspricht. Diese Spalte entspricht der Spalte deptno in der Tabelle dept. Diese Entsprechung muss innerhalb der Abfrage durch einen entsprechenden Ausdruck in einer Where-Klausel kenntlich gemacht werden (emp.deptno=dept.deptno). Die gewünschte Abfrage hierfür könnte folgendermaßen lauten:

```
select e.ename, d.dname
  from emp e, dept d
  where e.deptno = d.deptno
;
ENAME       DNAME
------------------------
SMITH       RESEARCH
ALLEN       SALES
```

```
WARD        SALES
JONES       RESEARCH
MARTIN      SALES
BLAKE       SALES
CLARK       ACCOUNTING
SCOTT       RESEARCH
KING        ACCOUNTING
TURNER      SALES
ADAMS       RESEARCH
JAMES       SALES
FORD        RESEARCH
MILLER      ACCOUNTING
```

Betrachten wir dies Anweisung etwas genauer. Als erstes fällt auf, dass hier mit Aliasnamen für die Tabellennamen gearbeitet wurde. Das ist vor allem bei Abfragen über mehrere Tabellen bzw. sehr langen Tabellennamen angenehmer und mit weniger Schreibarbeit verbunden. Aliasnamen werden vergeben, indem hinter den Tabellennamen in der From-Klausel einfach der Aliasname geschrieben wird. Da Abfragen nicht 'von links nach rechts' abgearbeitet werden, sondern ein konkreter Ausführungsplan erstellt wird, also die Abfrage als Ganzes ausgeführt wird, können die Aliasnamen überall in der Abfrage verwendet werden. Anstelle von der obigen Abfrage könnten Sie hier auch die folgende Abfrage schreiben.

```
select ename, dname from emp e, dept d where e.deptno = d.deptno;
```

schreiben, da die Spaltennamen in der Menge der verbundenen Tabellen eindeutig sind. Das heißt, es gibt in allen an diesem JOIN beteiligten Tabellen nur eine Spalte namen ename und auch nur eine Spalte namens dname. In der Praxis hat es sich jedoch primär aus Gründen der Übersichtlichkeit eingebürgert, auch bei eindeutigen Spaltennamen den jeweiligen Aliasnamen voranzustellen. Die Lesbarkeit wird hierdurch stark gesteigert. Würden Sie keine Alaisnamen für die beteiligten Tabellen verwenden, so könnte die Abfrage folgendermaßen aussehen:

```
select emp.ename, dept.dname from emp, dept
  where emp.deptno = dept.deptno
;
```

Auch hierbei könnten Sie aus den oben beschriebenen Gründen zwischen SELECT und FROM den Tabellennamen weglassen. Schauen wir uns den internen Ablauf der Abfrage etwas ausführlicher an. Zuerst wird der erste Datensatz aus der Tabelle emp extrahiert:

```
Empno ename ...  deptno
7369  Smith ...  20
```

Nun wird in der Tabelle dept nach der deptno 20 gesucht. Wird in der Tabelle der Datensatz mit der deptno 20 gefunden, so werden diese beiden Datensatze in der Ergebnismenge verbunden.

Tabelle emp
Tabelle dept
Empno ename ... deptno deptno dname loc
7369 Smith ... 20 20 research dallas

Da deptno der Primärschlüssel der Tabelle dept ist, braucht nicht weiter gesucht zu werden. Als nächsten Schritt wird der zweite Datensatz aus der Tabelle emp extrahiert.

```
Empno ename ...  deptno
7499 Allan ...  30
```

Nun wird in der Tabelle dept nach der deptno 30 gesucht. Wird in der Tabelle der Datensatz mit der deptno 30 gefunden, so werden diese beiden Datensatze in der Ergebnismenge verbunden.

Tabelle emp
Tabelle dept
Empno ename ... deptno deptno dname loc
7369 Smith ... 20 20 research dallas
7499 Allan ... 30 30 sales Chicago

Dieser Prozess durchläuft alle Datensätze der Tabelle emp. Wichtig hierbei ist, dass nur Datensätze, die tatsächlich in Beziehung stehen, in der Ergebnismenge auftauchen. In unserem Beispiel führt das dazu, dass die Abteilung mit der Nummer 40 in der Ergebnismenge gar nicht auftaucht, da es hierfür keinen Mitarbeiter gibt. Eine leere Abteilung sozusagen.

EQUIJOIN über mehr als zwei Tabellen

Nehmen wir einmal an, es gäbe eine weitere Tabelle, welche die Dienstwagentypen in Abhängigkeit der Position bzw. des Jobs darstellt.

Als Ergebnis wollen Sie den Namen des Mitarbeiters und der Abteilung, in der er arbeitet und zusätzlich noch den Dienstwagentyp ermitteln. Die Daten der Ergebnismenge erstrecken sich demnach über die drei Tabellen Mitarbeiter (emp), Abteilung (dept) und Dienstwagen (official_car). Die beiden Tabellen Mitarbeiter (emp) und Abteilung (dept) sind über die jeweilige Abteilungsnummer (deptno), die in beiden Tabellen steht, verbunden. Die beiden Tabellen Mitarbeiter (emp) und Dienstwagen (official_car) sind über den Job auf Seiten der Tabelle Mitarbeiter (emp) und über Position auf Seiten der Tabelle official_car. Wir haben folglich zwei WHERE-Kriterien, die beide gelten müssen. Infolgedessen werden diese beiden WHERE-Kriterien durch ein AND verbunden. Die Anweisung könnte folgendermaßen aussehen.

```
select d.dname,e.ename,e.job,oc.typ
  from official_car oc,emp e, dept d
  where oc.position=e.job and e.deptno=d.deptno
;
DNAME       ENAME       JOB         TYP
---------- ---------- ---------- --------------------
RESEARCH    JONES       MANAGER     BMW 7er
ACCOUNTING  CLARK       MANAGER     BMW 7er
SALES       BLAKE       MANAGER     BMW 7er
ACCOUNTING  KING        PRESIDENT   Mercedes SL
SALES       ALLEN       SALESMAN    BMW 3er
SALES       MARTIN      SALESMAN    BMW 3er
SALES       TURNER      SALESMAN    BMW 3er
SALES       WARD        SALESMAN    BMW 3er
```

Auch hier wird wieder ein Aliasname für die beteiligten Tabellen benutzt. Die Ergebnismenge könnte in unserem Beispiel folgendermaßen aussehen:

Vereinfacht kann man sagen, dass in der Regel eine Abfrage, die auf Daten von X verschiedenen Tabellen zugreift, X-1 verschiedene WHERE-Kriterien vonnöten sind. Zu beachten ist auch hier, dass die Jobs, denen keine Dienstwagen zustehen (Clerk etc.), auch nicht in der Ergebnismenge auftauchen.

Hinweis:
Bei der Anzahl der WHERE-Kriterien ist nicht entscheidend, wie viel Spalten ausgegeben werden. Nehmen wir uns das obere Beispiel. Angenommen, sie wollen lediglich wissen, welche Dienstwagen in welchen Abteilungen zur Verfügung stehen, nicht jedoch die Mitarbeiter. Auch in diesem Fall ist ein Join über alle drei Tabellen notwendig, um die richtige Ergebnismenge zu erhalten. Zwischen der Tabelle Abteilung und Dienstwagen allein gibt es kein zutreffendes WHERE-Kriterium. Somit würde auch in diesem Falle die Abfrage so aussehen:

```
select d.dname,oc.typ
  from official_car oc,emp e, dept d
  where oc.position = e.job and e.deptno = d.deptno
;
```

Der lieben Ordnung wegen wäre hier noch, um Duplikate bei der Ausgabe auszuschließen, ein DISTINCT das Mittel der Wahl:

```
select DISTINCT d.dname,oc.typ
  from official_car oc,emp e, dept d
  where oc.position = e.job and e.deptno = d.deptno
;
DNAME       TYP
-------------------------
ACCOUNTING  BMW 7er
ACCOUNTING  Mercedes SL
RESEARCH    BMW 7er
SALES       BMW 3er
SALES       BMW 7er
```

Non-Euqijoins

Wenn keine Spalte einer Tabelle mit einer Spalter einer anderen Tabelle direkt korrespondiert, dann ist nur ein Non-Euqijoin möglich.

OUTER JOIN

Bei einem OUTER JOIN werden nicht nur die Datensätze angezeigt, die in direkter Beziehung zueinander stehen, sondern auch Datensätze, die keinen direkten Bezug zu Datensätzen der anderen Tabelle haben. Als Beispiel betrachten wir die Tabellen Mitarbeiter (emp) und Abteilung (dept). Wie wir im vorigen Beispiel gesehen haben, gibt es keinen Mitarbeiter, der zur Abteilung Nummer 40 gehört. Dieser wird daher auch bei einem normalen JOIN nicht in der Ergebnismenge angezeigt. Es kann jedoch durchaus sein, dass Sie auch die Abteilung / Abteilungen in der Ergebnismenge sehen wollen, die keine direkte Beziehung zu einem Mitarbeiter haben, also die zur Zeit leeren Abteilungen. Genau hier kommt der OUTER JOIN zum Einsatz.

```
select e.ename,d.dname
  from emp e,dept d
  where e.deptno(+) = d.deptno
;
ENAME       DNAME
----------  ----------
CLARK       ACCOUNTING
KING        ACCOUNTING
MILLER      ACCOUNTING
SMITH       RESEARCH
ADAMS       RESEARCH
FORD        RESEARCH
SCOTT       RESEARCH
JONES       RESEARCH
ALLEN       SALES
BLAKE       SALES
MARTIN      SALES
JAMES       SALES
TURNER      SALES
WARD        SALES
            OPERATIONS
```

Durch das PLUS-Zeichen wird erreicht, dass alle Datensätze der dem PLUS gegenüberliegenden Tabelle in die Ergebnismenge aufgenommen werden, egal, ob es eine direkte Verbindung gibt oder nicht. Im Ergebnis können Sie hier erkennen, dass die Abteilung OPERATIONS keine Mitarbeiter hat. Würde sich das PLUS-Zeichen an der gegenüberliegenden Seite befinden, so würden auch die Mitarbeiter angezeigt werden, die zu keiner Abteilung gehören.

```
select e.ename,d.dname
  from emp e,dept d
  where e.deptno = d.deptno(+)
;
```

Mitarbeiter, die zu keiner Abteilung gehören, gibt es in unserem Beispiel jedoch nicht.

OUTER JOIN Anwendungen

Angenommen, Sie wollen herausfinden, in zu welcher Abteilung keine Mitarbeiter gehören. Es sollten jedoch nur diese Datensätze in der Ergebnismenge erscheinen. Folgende Abfrage würde das gewünschte Resultat bringen:

```
select e.ename,d.dname from emp e,dept d
  where e.deptno(+) = d.deptno AND
        e.empno IS NULL
;
ENAME     DNAME
--------------------
          OPERATIONS
```

Wenn wir uns die Abfrage etwas genauer anschauen, so erkennen wir, dass zuerst alle Datensätze aus der Tabelle Abteilung (dept) in die Ergebnismenge aufgenommen werden, egal, ob es eine direkte Verbindung zu Mitarbeitern gibt oder nicht.

```
ENAME      DNAME
----------------------
CLARK      ACCOUNTING
KING       ACCOUNTING
MILLER     ACCOUNTING
SMITH      RESEARCH
ADAMS      RESEARCH
FORD       RESEARCH
SCOTT      RESEARCH
JONES      RESEARCH
ALLEN      SALES
BLAKE      SALES
MARTIN     SALES
JAMES      SALES
TURNER     SALES
WARD       SALES
NULL       OPERATIONS
```

Für die Abteilungen, die keine Verbindung zu einem Mitarbeiter haben, sind alle dazugehörigen Spalten der Tabelle Mitarbeiter (emp) NULL. Somit müssen wir lediglich noch nach IS NULL bei den geeigneten Spalten der Tabelle Mitarbeiter (emp) einschränken. Als geeignete Spalten gelten alle Spalten, die nicht NULL sein können (als NOT NULL definiert wurden, siehe spätere Kapitel).

Kreuzprodukt

Unter einem Kreuzprodukt, auch Cartesian Product genannt, versteht man das Verbinden mehrerer Tabellen ohne Verbindungskriterium. Obgleich dies in den meisten Fällen 'sinnlos' erscheint, wollen wir hier ein kleines Beispiel vorstellen. Angenommen, sie veranstalten eine Schachrunde, in der jeder Mitarbeiter gegen jeden antreten muss. Folgende Abfrage, mit Hilfe eines Kreuzproduktes oder im englischen CROSS JOIN, würde die Partien ermitteln:

```
select e1.ename,e2.ename from emp e1, emp e2;
```

Diese Abfrage würde jeden Namen aus der Tabelle e1 (entspricht emp) mit jedem Namen aus der Tabelle e2(entspricht ebenfalls emp) verbinden und in der Ergebnismenge aufnehmen. Da die Tabelle emp 14 Datensätze hat, werden in der Ergebnismenge 14 * 14 Datensätze erscheinen, also 196. Da die Mitarbeiter schlecht gegen sich selbst spielen können, muss dieser Fakt noch ausgeschlossen werden.

```
select e1.ename,e2.ename from emp e1, emp e2
  where e1.ename <> e2.ename
;
```

Folglich sollten 196 weniger 14 Datensätze in der Ergebnismenge erscheinen. Jeder gegen jeden – mit Hin- und Rückrunde!

ACHTUNG:
Oftmals benutzt man auch versehentlich ein Kreuzprodukt – wenn Sie zum Beispiel vergessen haben, ein WHERE-Kriterium zu definieren und es mehr als eine Tabelle hinter FROM gibt. Sollten Sie zwei Tabellen mit beispielsweise 10000 Datensätzen verbinden (Kreuzprodukt), so hat die Ergebnismenge 10000 * 10000 Datensätze, also 100 Millionen. Übrigens auch ein gutes Mittel zum Erzeugen von großen Datenbeständen!

SELF JOIN

Unter einem SELF JOIN versteht man eine Verbindung einer Tabelle mit sich selbst. Dieser Join ist wohl der am schwersten zu verstehende JOIN, da eine Tabelle mit sich selbst verknüpft ist.

Um diesen SELF JOIN verständlich zu machen, benutzen wir das Beispiel der Tabelle Mitarbeiter (emp). In der Tabelle Mitarbeiter (emp) hat jeder Mitarbeiter eine Mitarbeiternummer empno und daneben noch die Mitarbeiternummer des Vorgesetzten (mgr).

Schauen wir uns hierfür die Tabelle etwas genauer an. Herr Smith hat die Mitarbeiternummer(empno) 7369. Sein Vorgesetzter ist der Herr mit der Mitarbeiternummer 7902 (entspricht der Spalte MGR), also der Herr Ford. Herr ALLAN hat die Mitarbeiternummer(empno) 7499. Sein Vorgesetzter ist der Herr mit der Mitarbeiternummer 7698 (entspricht der Spalte MGR), also der Herr Blake. Auf diese Weise wird die gesamte Hierarchie der Firma abgebildet. Soll nun in einer Abfrage ermittelt werden, welcher Mitarbeiter welchen Vorgesetzten (jeweils den Namen) hat, so ist die eine Verbindung der Spalten Mitarbeiternummer (empno) und der Spalte Manager (mgr) ein und derselben Tabelle:

```
select mitarbeiter.ename as "Mitarbeiter",
       vorgesetzter.ename as "Vorgesetzter"
  from
       emp mitarbeiter,
       emp vorgesetzter
  where mitarbeiter.mgr = vorgesetzter.empno
;
Mitarbeiter Vorgesetzt
----------------------
SMITH       FORD
ALLEN       BLAKE
WARD        BLAKE
JONES       KING
MARTIN      BLAKE
BLAKE       KING
CLARK       KING
SCOTT       JONES
TURNER      BLAKE
ADAMS       SCOTT
JAMES       BLAKE
FORD        JONES
MILLER      CLARK
```

Die Ergebnismenge würde folgendermaßen aussehen: Das einzige Manko ist, dass der Chef des Ganzen natürlich keine MGR hat – und demnach auch nicht angezeigt wird. Aber auch hier kann man sich leicht helfen. Wir nutzen einfach das Konzept der Outer Joins.

```
SELECT mitarbeiter.ename  AS "Mitarbeiter",
       vorgesetzter.ename AS "Vorgesetzter"
   FROM emp Mitarbeiter, emp vorgesetzter
   WHERE mitarbeiter.mgr = vorgesetzter.empno(+)
;
Mitarbeiter  Vorgesetzter
----------  ----------
SMITH       FORD
```

```
ALLEN       BLAKE
WARD        BLAKE
JONES       KING
MARTIN      BLAKE
BLAKE       KING
CLARK       KING
SCOTT       JONES
KING
TURNER      BLAKE
ADAMS       SCOTT
JAMES       BLAKE
FORD        JONES
MILLER      CLARK
```

Hierdurch wird erreicht, dass alle Mitarbeiter angezeigt werden, egal, ob diese auch einen Manager (Vorgesetzten) haben. Die Ergebnismenge würde nunmehr folgendermaßen ausschauen:

1.3.2 Mengenoperationen

MINUS

Von der ersten Menge wird die zweite Menge abgezogen. MINUS verwendet DISTINCT.

```
select field1, field2, ... field_n from tables
  MINUS
select field1, field2, ... field_n from tables;
```

```
select d.deptno from scott.dept d
  minus
select e.deptno from scott.emp e;
```

INTERSEC

INTERSEC ergibt die Überschneidungsmenge von Elementen, die zu beiden Mengen gehören.

```
select field1, field2, ... field_n from tables
  INTERSEC
select field1, field2, ... field_n from tables;
```

```
select distinct city from customers
  intersect
select distinct city from suppliers;
```

UNION

Ergibt die Gesamtmenge beider Mengen. UNION verwendet DISTINCT.

```
select field1, field2, ... field_n from tables
  UNION
select field1, field2, ... field_n from tables;
```

```
select supplier_id from suppliers
  UNION
select supplier_id from orders;
```

UNION ALL

Ergibt die Gesamtmenge beider Mengen. Im Gegensatz zu UNION wird kein DISTINCT verwendet.

```
select field1, field2, ... field_n from tables
  UNION ALL
select field1, field2, ... field_n from tables;

select supplier_id from suppliers
  UNION ALL
select supplier_id from orders;
```

1.3.3 Übungen 04

Übungen siehe Seite 241.

1.3.4 Gruppierungen

Oftmals sollen nicht einzelne Werte zurückgegeben werden, sondern Berechnungen über Gruppen von Werten hinweg.

SUM()

```
SELECT sum(sal) AS "Summe der Gehälter" FROM emp;
Summe der Gehälter
------------------
             29025
```

Ermittelt wird die Summe der Gehälter aller Mitarbeiter.

MIN()

```
select min(sal) AS "Kleinstes Gehalt" from emp;
Kleinstes Gehalt
----------------
             800
```

Ermittelt wird das kleinste Gehalt aller Mitarbeiter.

MAX()

```
SELECT max(sal) AS "Größtes Gehalt" FROM emp;
Größtes Gehalt
--------------
         5000
```

Ermittelt wird das größte Gehalt aller Mitarbeiter.

AVG()

```
SELECT avg(sal) AS "Durchschnittsgehalt" FROM emp;
Durchschnittsgehalt
-------------------
         2073,21429
```

Ermittelt wird das Durchschnittsgehalt aller Mitarbeiter.

COUNT()

```
SELECT COUNT(EMPNO) AS "Mitarbeiteranzahl" FROM emp;
MITARBEITERANZAHL
-----------------
               14
```

Ermittelt wird die Anzahl der Einträge in der Spalte empno. Es wird also die Mitarbeiteranzahl ermittelt. Zu beachten ist hierbei, dass als COUNT-Kriterium eine Spalte genommen werden sollte, die nicht leer sein kann. Wird als COUNT-Kriterium eine NULL-Spalte benutzt, wird diese nicht mitgezählt. Wenn diese Abfrage z.B. wie folgt wäre, so wäre das Ergebnis lediglich 4:

```
SELECT COUNT(comm) AS "Mitarbeiteranzahl" FROM emp;
MITARBEITERANZAHL
-----------------
                4
```

In 10 Datensätzen steht nämlich NULL in der Spalte comm.

GROUP BY – Gruppen bilden

Mit der SQL-Anweisung GROUP BY werden Datensätze zu Gruppen zusammengefasst. Die Gruppenfunktionen gelten dann lediglich für die einzelnen Gruppen. Eine Gruppe ist durch die Identität der Inhalte gekennzeichnet. Das bedeutet, wenn nach der Abteilungsnummer deptno in der Tabelle Mitarbeiter (emp) mit GROUP BY gruppiert wird, so werden jeweils Gruppen mit den gleichen Abteilungsnummern gebildet und die entsprechende Gruppenfunktion auf die gebildeten Gruppen angewendet.

```
SELECT deptno,avg(sal) as "Durchschnittsgehalt je Abt"
  FROM EMP group by deptno
;
DEPTNO  Durchschnittsgehalt je Abt
---------------------------------
10              2916,66667
20              2175
30              1566,66667
```

Durch die Gruppierung GROUP BY deptno werden jeweils Gruppen mit identischen Abteilungsnummern gebildet. Über diese Gruppen wird nun die Funktion ausgeführt, also der Durchschnitt der Gehälter ermittelt.

HAVING – Einschränkungen nach Gruppenfunktionen

Häufig sollen nicht alle Gruppen ausgegeben werden. Wenn Sie nach Gruppenfunktionen einschränken wollen, so benötigen Sie die Anweisung HAVING.

```
SELECT deptno,avg(sal) as "Durchschnittsgehalt je Abt"
  FROM EMP group by deptno
    HAVING avg(sal) > 2000
;
DEPTNO  Durchschnittsgehalt je Abt
-----------------------------------
   10              2916,66667
   20               2175
```

Auch hier wird das Durchschnittsgehalt je Abteilung ermittelt. Jedoch nur die Durchschnittsgehälter, die über 2000 liegen, werden ausgegeben.

Besonderheiten bei NULL und AVG

Die Funktion AVG zieht in die Berechnung des Durchschnittes lediglich die Zeilen ein, die nicht NULL in der entsprechenden Spalte sind. Dies führt zu folgendem Ergebnis:

```
select avg(comm) from emp;
AVG(COMM)
---------
   550
```

Es gibt jedoch nur 4 Datensätze, in denen die Spalte Comm nicht NULL ist. In diesen Datensätzen ist die Spalte Comm 300, 500, 1400 und 0. Das Ergebnis von 550 ergibt sich demnach aus (300+500+1400+0) / 4. Sollen jedoch auch die Datensätze einbezogen werden, in denen Comm NULL ist, so können Sie diese mit der Funktion NVL durch eine 0 ersetzen lassen.

```
select avg(nvl(comm,0)) from emp;
AVG(NVL(COMM,0))
----------------
   157,142857
```

Durch die Funktion NVL(comm,0) werden NULL-Werte in der Spalte comm als 0 interpretiert. Demnach wird nun (300+500+1400+0+0+0+0+0+0+0+0+0+0+0) / 14 gerechnet.

1.3.5 Übungen 03

Übungen siehe Seite 239.

1.3.6 Unterabfragen

Häufig wollen Sie nicht das geringste Gehalt als Ergebnismenge, sondern den oder die Mitarbeiter, die das geringste Gehalt erhalten. Hierfür leistet eine Unterabfrage gute Dienste.

```
select ename from emp where sal =
  (
    select min(sal) from emp
  )
;
```

In der in Klammern eingeschlossenen Unterabfrage wird zuerst das geringste Gehalt ermittelt (z.B. 800). Nun wird für die Unterabfrage der ermittelte Wert eingesetzt. Somit lautet die Abfrage:

```
select ename from emp where sal = 800;
ENAME
------
SMITH
```

Hierdurch wird der Name des oder der Mitarbeiter zurückgegeben, der diese 800 verdient – die natürlich wiederum dem geringsten Gehalt entsprechen.

Inline View

Inline Views sind Unterabfragen nach FROM.

```
select  a.last_name, a.salary, a.department_id, b.maxsal
    FROM employees a,
      (
        SELECT department_id,max(salary) maxsal
          FROM employees
          GROUP BY department_id
      ) b
      WHERE a.department_id = department_id
        AND a_salary < b.maxsal
;
```

IN

Gibt die Unterabfrage mehr als einen Wert zurück, so können Sie natürlich nicht auf Gleichheit oder Ungleichheit testen. Hier müssen Sie mit entsprechenden Operatoren, wie IN arbeiten.

```
select distinct d.dname from dept d
  where d.deptno in
   (
     select distinct deptno from emp
       where job = 'SALESMAN' or job = 'MANAGER'
   )
;
DNAME
--------------
```

```
ACCOUNTING
RESEARCH
SALES
```

In der Unterabfrage werden die Abteilungsnummern ermittelt, in denen Mitarbeiter der Positionen SALESMAN oder MANAGER arbeiten. In der äußeren Abfrage werden hierzu die Abteilungsnamen ausgegeben. Würden Sie hier anstelle des IN ein = benutzen, so würde ein Fehler ausgegeben werden:

```
FEHLER in Zeile 1:
ORA-01427: Unterabfrage für eine Zeile liefert mehr als eine Zeile
```

ANY

Einer der Datensätze muss der Bedingung genügen.

```
select * from employees where employeeid > any
  (select employeeid from orders)
;
```

ALL

Alle Datensätze muss der Bedingung genügen (AND).

```
select * from employees where employeeid > all
  (select employeeid from orders)
;
```

1.3.7 Top-N Analyse

```
select ROWNUM AS Rank, Name, Region, Sales from
  (
    select Name, Region, sum(Sales) AS Sales from Sales
   GROUP BY Name, Region
   order by sum(Sales) DESC
  )
  WHERE ROWNUM <= 10
;
```

Welche drei Mitarbeiter verdienen am besten?

```
select * from
  (
    select * from scott.emp order by SAL desc
  )
  where ROWNUM < 4
;
```

1.3.8 Übungen 05

Übungen siehe Seite 245.

1.4 Data Definition Language (DDL)

1.4.1 CREATE TABLE

Sie können eine Tabelle auf unterschiedliche Art und Weise erstellen. Als erstes schauen wir uns die konservative Methode mit der CREATE TABLE Anweisung an.

```
create table tablename
  (spalte 1 Datentyp, Spalte 2 Datentyp ... Spalte n Datentyp)
;
```

Folgende Regeln für die Namensvergabe sind zu beachten:

- Das erste Zeichen muss ein Buchstabe sein.
- Die Länge kann nicht 30 Zeichen überschreiten.
- Verwendbar sind die Zeichen A-Z, a-z, 0-9, _, $ und #
- Im gelichen Schema kann ein Name nur einmal vergeben werden.
- Ein Oracle-reserviertes Wort darf nicht verwendet werden.

```
create table produkt (Produktnummer number(10), Produktname char(30));
```

Es wird eine Tabelle namens Produkt erzeugt, welche aus zwei Spalten besteht.

Wichtige Datentypen:

```
Number    (Gesamtstellen,Nachkommastellen)
Varchar2  (Zeichenzahl)
Date
```

Eine Übersicht aller Datentypen befindet sich in [13].

Erstellen von Tabellen auf Basis einer Abfrage

Oftmals ist es notwendig, eine neue Tabelle zu erstellen, die auf Daten einer oder mehrerer bereits bestehender Tabellen aufbaut. Hierfür können Sie die neue Tabelle auf Basis einer Unterabfrage erstellen:

```
create table tablename as Unterabfrage;
```

```
create table emp_from_dept_10 as
  select empno,ename from emp where deptno = 10
;
```

Es wird eine neue Tabelle erstellt, welche genau die Datensätze enthält, die der Abfrage entsprechen. Die Datentypen richten sich nach den Datentypen der Spalten in der oder den Originaltabellen.

```
select * from emp_from_dept_10
EMPNO  ENAME
---------------
 7782  CLARK
 7839  KING
 7934  MILLER
```

Es ist ebenfalls möglich, eine neue Tabelle auf Grundlagen einer Abfrage über mehrere Tabellen zu erstellen.

```
create table emp_from_dept_10 as
  select e.empno, e.ename, d.dname from emp e, dept d
    where e.deptno = d.deptno and d.deptno = 10
;
```

Es wird eine neue Tabelle, welche genau die Datensätze enthält, die der Abfrage entsprechen. Zusätzlich wird noch der Name der Abteilung angezeigt.

```
select * from emp_from_dept_10
EMPNO  ENAME    DNAME
---------------------------
 7782  CLARK    ACCOUNTING
 7839  KING     ACCOUNTING
 7934  MILLER   ACCOUNTING
```

1.4.2 DROP TABLE

```
DROP TABLE tabellenname;
```

Alle Daten und die Struktur der Tabelle werden gelöscht. Alle offenden Transaktionen werden committet (automatisches COMMIT). Alle zugehörigen Indezes werden gelöscht. Alle auf dieser Tabelle basierenden VIEWs und SYNONYMs werden nicht gelöscht. DROP kann nicht per ROLLBACK rückgängig gemacht werden.

1.4.3 TRUNCATE

Mit der Anweisung TRUNCATE können Sie alle Datensätze einer Tabelle sehr schnell löschen. Allerdings können diese Datensätze dann mit einem ROLLBACK nicht mehr zurückgerollt werden.

```
truncate table tabellenname
```

```
truncate table emp;
```

Alle Datensätze der Tabelle Mitarbeiter (emp) werden gelöscht.

1.4.4 ALTER TABLE

Mit der Anweisung ALTER TABLE ist es möglich, eine bestehende Tabellenstruktur zu verändern. Sie können bestehende Spalten verändern, neue Spalten hinzufügen und Spalten bei Bedarf auch löschen.

```
alter table emp add birthdate date;
DESC emp;
Name      Null?     Typ
----------------------------------------
EMPNO     NOT NULL  NUMBER(4)
ENAME               VARCHAR2(10)
JOB                 VARCHAR2(9)
MGR                 NUMBER(4)
HIREDATE            DATE
SAL                 NUMBER(7,2)
COMM                NUMBER(7,2)
DEPTNO              NUMBER(2)
BIRTHDATE           DATE
```

Es wird zur Tabelle emp eine neue Spalte namens birthdate und dem Datentyp date hinzugefügt.

```
alter table emp drop column birthdate;
```

Es wird die Spalte birthdate einschließlich der Inhalte gelöscht.

```
DESC emp;
Name       Null?     Typ
-----------------------------------
EMPNO      NOT NULL  NUMBER(4)
ENAME                VARCHAR2(10)
JOB                  VARCHAR2(9)
MGR                  NUMBER(4)
HIREDATE             DATE
SAL                  NUMBER(7,2)
COMM                 NUMBER(7,2)
DEPTNO               NUMBER(2)


ALTER TABLE emp MODIFY job NOT NULL;
DESC emp;
Name       Null?     Typ
---------------------------------
EMPNO      NOT NULL  NUMBER(4)
ENAME                VARCHAR2(10)
JOB        NOT NULL  VARCHAR2(9)
MGR                  NUMBER(4)
HIREDATE             DATE
SAL                  NUMBER(7,2)
COMM                 NUMBER(7,2)
DEPTNO               NUMBER(2)
```

Die Spalte job wird so verändert, dass NULL-Werte in dieser Spalte nicht mehr erlaubt sind.

```
alter table emp modify job varchar2(20);
DESC emp;
Name        Null?    Typ
----------------------------------------
EMPNO       NOT NULL NUMBER(4)
ENAME                VARCHAR2(10)
JOB                  VARCHAR2(20)
MGR                  NUMBER(4)
HIREDATE             DATE
SAL                  NUMBER(7,2)
COMM                 NUMBER(7,2)
DEPTNO               NUMBER(2)
```

Es wird der Datentyp (genauer der Wertebereich) der Spalte job verändert

Zusammenfassung:

- Die Größe einer Spalte kann immer vergrößert werden.
- Eine NULL-Spalte kann immer hinzugefügt werden.
- Eine Spalte, die noch keine Werte hat, kann verkleinert werden und der Datentyp kann ebenfalls verändert werden.
- Eine NOT NULL Spalte kann hinzugefügt werden, wenn die gesamte Tabelle leer ist.
- Besitzt eine Spalte in jedem Datensatz einen Wert, kann diese Spalte in eine NOT NULL Spalte umgewandelt werden.

1.4.5 CREATE VIEW

Eine Sicht ist genau genommen nichts weiter als eine gespeicherte Abfrage. Diese gespeicherte Abfrage wird wie eine Tabelle behandelt. Sie kann genau wie eine Tabelle abgefragt werden. Selbst Einfüge-, Änderungs- oder Löschoperationen sind unter bestimmten Umständen möglich. Dies hängt von der gespeicherte Abfrage ab.

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view [(alias[, alias] ...)]
  AS subquery
  WITH CHECK OPTION [CONSTRAINT constraint]]
  WITH READ ONLY;
```

OR REPLACE – Erzeugt die View neu wenn diese bereits existiert.
FORCE – Erzeugt die View, auch wenn die Tabelle nicht existert.
WITH CHECK OPTION – Nur die Datensätze, die in der View angezeigt werden, können verändert werden.
WITH READ ONLY – Kein DML-Befehl kann auf diese View angewendet werden.

Der eigentliche Nutzen einer Sicht ist das Verstecken einer recht komplizierten und umfangreichen Abfrage. So ist es einfacher, alle Datensätze der Sicht research_mitarbeiter abzufragen als sich die Abfrage, welche alle Research-Mitarbeiter ermittelt, zu schreiben und anschließend auszuführen.

Ein weiterer Vorteil einer Sicht ist die Möglichkeit, Berechtigungen auf Sichtebene anzugeben. Es ist durch Sichten demnach möglich, z.B. dem Datenbankbenutzer anna die Berechtigung zu geben, sich alle Datensätze der Mitarbeiter

(emp) anzeigen zu lassen, außer die Datensätze der Mitarbeiter der Research-Abteilung. Hierfür brauchen Sie nur eine kleine Sicht, die alle Mitarbeiter mit Ausnahmen der Research-Mitarbeiter auswählt. Nun geben Sie Anna die notwendigen Berechtigungen auf die Sicht.

```
create view Research_Mitarbeiter as
  select * from emp
    where deptno=
     (select deptno from dept where dname = 'RESEARCH')
;
```

Es wird ein View erstellt, der alle Spalten der Tabelle Mitarbeiter (emp) selektiert, die aus der Abteilung RESEARCH kommen.

1.4.6 Ändern einer View

Das Ändern einer View erfolgt mit der Option OR REPLACE:

```
create or replace view Research_Mitarbeiter as
  select * from emp
    where deptno=
     (select deptno from dept where dname = 'SALES')
;
```

1.4.7 DROP VIEW

```
drop view Research_Mitarbeiter;
```

1.4.8 CREATE [PUBLIC] SYNONYM

Ein SYNONYM ist ein alternativer Name für eine Tabelle, einen VIEW oder gar für ein anderes SYNONYM. Synonyme dublizieren – genauso wie Views – keine Daten, sondern stellen nur eine zusätzliche Form bereit, um dieselben Daten darzustellen.

Ein PUBLIC-SYNONYM (öffentliches Synonym) kann von jedem benutzt werden.

```
create public synonym d for scott.dept;
```

1.4.9 DROP SYNONYM

```
drop synonym d;
```

1.4.10 CREATE SEQUENCE

Oftmals ist es hilfreich, einen Zähler in einer Tabelle zu verwenden. Zuerst muss die Sequence erzeugt werden. Danach kann auf die Sequence in der oder den Tabellen zugegriffen werden.

Mit der Pseudocoloumn `sequencename.nextval` können Sie den nächsten Wert der Sequence erzeugen und in die Tabelle einfügen. Nehmen wir an, Sie haben folgende leere Tabelle erzeugt:

```
create sequence s1 start with 1 maxvalue 5 increment by 1 nocycle;
```

Es wird eine Sequence namens s1 erzeugt. Der Startwert dieser Sequence ist 1 (start with 1) und der Maximalwert beträgt 5 (maxvalue=5). Erhöht wird jeweils um 1 (increment by 1). Wird der Maximalwert erreicht, so wird nicht von vorn begonnen (nocycle).

```
insert into produkt(produktnummer,produktname)
             values(s1.nextval,'Milch');
insert into produkt(produktnummer,produktname)
             values(s1.nextval,'Butter');
insert into produkt(produktnummer,produktname)
             values(s1.nextval,'Brot')
;
select * from produkt;
PRODUKTNUMMER  PRODUKTNAME
---------------------------
            1  Milch
            2  Butter
               Brot
```

Es wurden drei Produkte hinzugefügt und eine neue Produktnummer jeweils über s1.nextval erzeugt. Dies geht bei der oben erstellten Sequence s1 bis zur Zahl 5, danach muss beim Einfügen von Datensätzen wieder ein Wert eingegeben werden.

```
create sequence product_seq start with 1 increment 1;
insert into t_emp values (product_seq.nexval, 'Joe Black');
```

Mit der Funktion `sequencename.currval` können Sie sich den aktuellen Wert der Sequence anschauen.

```
select s1.currval from dual;
CURRVAL
-------
      3
```

Es wird er aktuelle Wert der Sequence s1 ermittelt und zurückgegeben

Wichtige Optionen der Create Sequence Anweisung

`increment by n`	Wert, um den jeweils erhöht wird (Standard ist 1).
`start with n`	erste zu generierende Sequencenummer
`maxvalue`	Höchstwert der Sequence.
`minvalue`	Mindestwert der Sequence.
`cycle sequence`	fängt nach Erreichen des Maximalwertes von vorn an (ist nicht Standard).
`nocycle sequence`	fängt nach Erreichen des Maximalwertes nicht von vorn an. (Ist Standard - vergleiche mit Cycle)
`cache n`	Es werden n Sequencewerte im Speicher gehalten.
`nocache`	Es werden keine Sequencewerte im Speicher gehalten.

Informationen über Sequences mit Hilfe des Data Dictionary

```
select sequence_name,min_value,max_value,increment_by,last_number
  from user_sequences
;
SEQUENCE_NAME MIN_VALUE MAX_VALUE INCREMENT_BY LAST_NUMBER
------------------------------------------------------------
S1                    1         5            1           6
```

Mit dieser Sicht des Data Dictionary werden die Optionen der für die Sequences ermittelt. last_number ist der nächste verfügbare Sequencewert.

Löschen einer Sequence

```
drop sequence Sequencename ;
```

1.4.11 INDEX

Nutzen eines Index

Einen Index können Sie sich wie ein Telefonbuch vorstellen. Wenn Sie einen Namen suchen, schlagen Sie das Telefonbuch zuerst in etwa in der Mitte auf und vergleichen den Namen, den Sie dort vorfinden, mit dem von Ihnen gesuchten Namen. Ist der von Ihnen gesuchte Name größer (weiter hinten im Alphabet), so können Sie die vordere Hälfte getrost vergessen – denn der Name muss sich in der hinteren Hälfte befinden. Nun schlagen Sie die Hälfte der hinteren Hälfte auf u.s.w.

Haben Sie dann endlich die Seite gefunden, auf der sich der von Ihnen gesuchte Name befindet, lesen Sie die Seite von Anfang bis Ende – bis Sie den Namen gefunden haben. Nun können Sie beruhigt telefonieren. Wäre das Telefonbuch nicht alphabetisch sortiert, so müssten Sie das ganze Buch durchblättern, um die entsprechende Telefonnummer herauszubekommen.

Ein Index arbeitet ähnlich dem Telefonbuchprinzip. Zuerst wird die indizierte Spalte sortiert und anschließend als Baumstruktur gespeichert. Auf die unterschiedlichen Speicherformen von Indizes soll an dieser Stelle verzichtet werden.

CREATE INDEX

```
create index indexname on tabelle(spalte1, spalte2, ... spalte n);
```

```
create index id1 on emp(ename);
```

Es wird ein Index namens id1 erstellt, der sich über die Spalte ename der Tabelle emp erstreckt. Hierdurch wird das Suchen nach ename beschleunigt. Doppelt vorkommende ename sind erlaubt.

```
create unique index id1 on emp(ename);
```

Es wird ein Index namens id1 erstellt, der sich über die Spalte Ename der Tabelle emp erstreckt. Doppelt vorkommende ename sind nicht erlaubt.

Automatische INDEX-Erstellung

Bei den Constraints PRIMARY KEY und UNIQUE wird automatisch ein Index erstellt.

DROP INDEX

```
drop index indexname
```

```
drop index id1
```

Der Index namens id1 wird gelöscht.

Informationen über bestehende Indizes

Mit Hilfe bestimmter Data Dictionary Table und Sichten erhalten Sie eine Übersicht über die bestehenden Indizes und die beteiligten Tabellen. In der Data Dictionary Sicht **user_indexes** sind alle Indizes und die Eindeutigkeit gespeichert. In der Data Dictionary Sicht **user_ind_columns** stehen die Tabellennamen und die entsprechenden beteiligten Spalten.

```
SELECT i1.index_name, i2.column_name
  FROM user_indexes i1, user_ind_columns i2
  WHERE i1.index_name = i2.index_name
    AND i2.table_name = 'EMP'
;
Index_name  Column_name
------------------------
PK_EMP      EMPNO
ID1         ENAME
```

Es werden alle Indizes ermittelt, die sich auf die Tabelle emp beziehen. Ausgegeben wird der Indexname und die beteiligte Spalte bzw. die beteiligten Spalten

Richtlinien für das Erstellen eines Index

Indizes machen nur Sinn, wenn diese richtig und maßvoll eingesetzt werden. Ein Index dient primär dazu, die Suchoperationen zu beschleunigen. Allerdings muss ein Indexeintrag auch immer mitgepflegt werden. So viel Indizes wie nötig – nicht mehr!

Wann sollten Sie einen Index erstellen?

-
- Die Spalte wird häufig als Join-Kriterium benutzt.
- Die meisten Abfragen ermitteln als Ergebnismenge weniger als 5% der Gesamtmenge.
- Die Tabelle enthält einen sehr selektiven Wertebereich, dass heißt, es kommen verhältnismäßig wenig doppelte Spaltenwerte vor.

Wann sollten Sie keinen Index erstellen?

- Die Tabelle ist eher klein.
- Die Tabelle wird häufig aktualisiert.
- Die entsprechende Spalte wird selten in Suchfunktionen benutzt.
- Die Spalte ist nur wenig selektiv (z.B. Anrede o. ä.).

1.5 Constraints

1.5.1 Primary Key

Erstellen eines Primary Key

Jede Tabelle besitzt eine Spalte oder zumindest eine Spaltenkombination, die eindeutig ist. Das bedeutet, mit dieser Spaltenkombination ist es möglich, jeden Datensatz der Tabelle eindeutig von den anderen Datensätzen abzugrenzen. Diese Spalte bzw. Spaltenkombination wird als Primärschlüssel bezeichnet und darf nicht doppelt innerhalb einer Tabelle auftauchen. Der Primärschlüssel darf nicht NULL sein. Bei einem zusammengesetzten Primärschlüssel darf keine der am Primärschlüssel beteiligten Spalten NULL-Werte beinhalten.

Im Zuge des Erstellens eines Primärschlüssel wird automatisch von Oracle auf diese Spalte bzw. Spaltenkombination ein eindeutiger Index erstellt.

```
alter table emp add constraint PK1 primary key (empno);
```

In der Tabelle Mitarbeiter (emp) wird ein neuer Primärschlüssel erstellt, der die Spalte Mitarbeiternummer (empno) beinhaltet. Dies ist natürlich nur dann möglich, wenn es noch keinen Primärschlüssel in dieser Tabelle gibt

Löschen eines Primary Key

```
ALTER TABLE tabellenname DROP CONSTRAINT Constraintname;
```

```
ALTER TABLE employees DROP CONSTRAINT empl_pk;
```

1.5.2 Foreign Key

Der Foreign Key Constraint ist ein Constraint unter SQL, dessen Aufgabe darin besteht, die referentielle Integrität von zwei Tabellen sicherzustellen.

```
ALTER TABLE tabellenname
  ADD CONSTRAINT constraintname
  FOREIGN KEY (spalte1, spalte2 ...)
  REFERENCES tabellenname(spalte1, spalte2 ...)
;
```

Nehmen wir hierfür das Beispiel der beiden Tabellen Mitarbeiter (emp) und Abteilung (dept). Die beiden Tabellen stehen in einer 1:n Beziehung zueinander. Eine Abteilung besteht aus mehreren Mitarbeitern bzw. kann aus mehreren Mitarbeitern bestehen. Ein Mitarbeiter wiederum ist in genau einer Abteilung.

Die Verbindung der beiden Tabellen wird über die in beiden Tabellen vorkommende Spalte Abteilungsnummer (deptno) hergestellt. Ein Mitarbeiter, der in einer Abteilung arbeitet, die es gar nicht gibt, macht keinen Sinn. Daher dürfen in der Tabelle Mitarbeiter nur Abteilungsnummern stehen, die auch in der Tabelle Abteilung auftauchen.

Weiterhin können nur Abteilungen gelöscht werden, in denen keine Mitarbeiter mehr arbeiten. Wäre dies nämlich möglich, hätten wir wieder das Problem, dass es Mitarbeiter gibt, die in Abteilungen arbeiten, die es gar nicht mehr gibt.

Oracle bietet dennoch eine Löschmöglichkeit für die Tabelle Abteilung. Beim kaskadierenden Löschen werden alle Mitarbeiter, die in der gelöschten Abteilung arbeiteten, mitgelöscht. Die eben beschriebenen Merkmale der referentiellen Integrität werden erst durchgesetzt, nachdem ein Foreign Key Constraint erstellt wurde.

```
alter table emp add constraint FK1
  foreign key (deptno) references dept(deptno)
;
```

Ein Foreign Key Constraint wird immer in der Tabelle der n-Seite erstellt und referenziert auf die entsprechende Spalte der 1-Seite. Die Spalte, auf die referenziert wird, muss als UNIQUE und NOT NULL definiert sein. Ein Foreign Key Constraint kann auch eine Spaltenkombination beinhalten. Hierbei werden die in den Klammern eingeschlossenen Spalten durch ein Komma getrennt!

Wenn Sie diesen Foreign Key Constraint erstellt haben, so wird die referentielle Integrität zwischen den beiden Tabellen gewahrt bzw. erzwungen. Versuchen Sie, in die Tabelle Mitarbeiter (emp) einen Datensatz hinzuzufügen, der eine Abteilungsnummer besitzt, die es in der Tabelle Abteilung (abt) nicht gibt, so führt dies zu einer Verletzung des Foreign Key Constraint und es erfolgt folgende

```
insert into emp (empno,ename,deptno) values(9999,'Mueller',50);
ORA-02291:
Verstoß gegen Integritätsregel (SCOTT.FK1).
Übergeordn. Schlüssel nicht gefunden
```

Wenn Sie versuchen, die Abteilung 30 zu löschen, wird auch hierbei die referentielle Integrität verletzt.

```
delete from dept where deptno=30;
ORA-02292:
Verstoß gegen Integritätsregel (SCOTT.FK1).
Untergeordneter Datensatz gefunden.
```

Hinweis: Beschreibungen zu den ORA-Fehlern können Sie mit dem Programm oerr erhalten:

```
host
oerr ora 02291
02291, 00000,"integrity constraint (%s.%s) violated - parent key not found"
// *Cause: A foreign key value has no matching primary key value.
// *Action: Delete the foreign key or add a matching primary key.
exit
```

Wenn Sie versuchen, die Abteilung 40 zu löschen, wird hierbei die referentielle Integrität nicht verletzt, da es keine Mitarbeiter gibt, die zur Abteilung 40 gehören.

Wenn Sie versuchen, die Abteilung 30 in Abteilung 45 umzubenennen, wird auch hierbei die referentielle Integrität verletzt.

```
update dept set deptno=45 where deptno=30;
ORA-02292:
Verstoß gegen Integritätsregel (SCOTT.FK1).
Untergeordneter Datensatz gefunden.
```

Zum Abschluss der Foreign Key Constraint wollen wir uns die Möglichkeit des **kaskadierenden Löschens** anschauen. Hierfür löschen wir den Constraint FK1 und erstellen diesen mit der Option ON DELETE CASCADE neu.

```
alter table emp drop constraint fk1;
alter table emp add
  constraint fk1
  foreign key (deptno)
  references dept(deptno) ON delete cascade
;
```

Zuerst wird der Foreign Key Constraint FK1 gelöscht und anschließend mit der Option ON DELETE CASCADE neu erstellt. Sehen wir uns jetzt an, was im Falle des Löschens einer Abteilung passiert:

```
delete from dept where deptno = 30;
```

Eine Zeile wurde gelöscht. Trotz referentieller Integrität wurde eine Zeile (deptno = 30) aus der Tabelle Abteilung gelöscht. Des weiteren wurden alle Mitarbeier, die zur Abteilung 30 gehörten, ebenfalls gelöscht. Hiervon können Sie sich gern durch folgende Abfrage überzeugen.

```
select * from emp where deptno = 30;
```

1.5.3 CHECK

Mit einem Check-Constraint wird festgelegt, welche Bedingung bzw. Bedingungen jeder Datensatz der Tabelle erfüllen muss. Es kann je Tabelle mehrere Check-Constraints geben, die dann natürlich alle eingehalten werden müssen. Check-Constraint dienen zur Wahrung der Integrität der Daten und erweitern die durch Datentypen zur Verfügung gestellten Integritätsmöglichkeiten.

```
ALTER TABLE tabellenname
  ADD CONSTRAINT Constraintname CHECK (bedingung)
;
```

Die Anwendung eines Check-Constraint wollen wir am Beispiel der Tabelle Mitarbeiter (emp) verdeutlichen. Wir nehmen an, dass die Spalte sal nicht größer als 100000 sein darf.

```
alter table emp add constraint ck1 check (sal < 100000);
```

Zur Tabelle emp wird ein Check-Constraint namens ck1 hinzugefügt. Dieser Check-Constraint überprüft, ob das Gehalt (sal) kleiner als 100000 ist. Ist dies nicht der Fall, erscheint folgende

```
insert into emp (empno,ename,sal) values(9999,'Mueller',110000);
FEHLER in Zeile 1:
ORA-01438:
Wert größer als angegebene Stellenzahl für diese Spalte zulässt
```

Unterabfragen sind in Check-Constraint nicht möglich:

```
alter table emp add constraint ck2 check
  (sal<(select max(sal) from emp))
;
FEHLER in Zeile 1:
ORA-02251:
Unterabfrage hier nicht zulässig
```

Hierdurch sollte erreicht werden, dass kein Gehalt eingefügt werden darf, welches größer als das bisher höchste Gehalt ist.

Durch einen Check-Constraint wird die gewünschte Einschränkung gewährleistet. Dies gilt selbstverständlich auch für etwaige Updates.

```
update emp set sal = 110000;
FEHLER in Zeile 1:
ORA-01438:
Wert größer als angegebene Stellenzahl für diese Spalte zulässt.
```

Das Gehalt aller Mitarbeiter soll auf 110000 erhöht werden. Dieser Wert ist größer als 100000 und verstößt demnach gegen den Check-Constraint.

Löschen eines CHECK Key

```
ALTER  TABLE tabellenname DROP CONSTRAINT Constraintname;
```

```
ALTER  TABLE employees DROP CONSTRAINT empl_CK1;
```

1.5.4 UNIQUE

Erstellen

Durch einen Unique-Constraint können Sie sicherstellen, dass Werte in bestimmten Spalten bzw. Spaltenkombinationen eindeutig sind und bleiben. NULL-Werte sind hierbei in den beteiligten Spalten erlaubt.

Bei UNIQUE wird automatisch ein INDEX erstellt.

```
ALTER TABLE tabellenname
  ADD CONSTRAINT Constraintname
  UNIQUE (spalte1,spalte2...)
;
```

```
alter table dept add constraint u1 unique (dname);
```

Es wird ein Unique-Constraint über die Spalte dname erstellt. Es ist nun nicht mehr möglich, in der Tabelle dept zwei mal den gleichen Abteilungsnamen anzugeben. Wird dies versucht, erscheint ein entsprechender Fehler:

```
insert into dept (deptno, dname) values(50,'RESEARCH');
ORA-00001:
Verstoß gegen Eindeutigkeit, Regel (SCOTT.U1)
```

Durch die INSERT-Anweisung wird versucht, in die Tabelle Abteilung eine Abteilungsnamen hinzuzufügen, den es bereits gibt. Dies wird durch den UNIQUE-Constraint verhindert.

```
update dept set dname='RESEARCH' where deptno=10;
ORA-00001:
Verstoß gegen Eindeutigkeit, Regel (SCOTT.U1)
```

Durch die UPDATE-Anweisung wird versucht, in die Tabelle Abteilung einen Abteilungsnamen auf einen anderen Abteilungsnamen zu ändern, den es bereits gibt. Dies wird durch den Unique-Constraint verhindert.

Löschen

```
ALTER TABLE tabellenname DROP CONSTRAINT Constraintname;
```

```
ALTER TABLE employees DROP CONSTRAINT empl_UQ1;
```

1.5.5 NOT NULL

Durch den NOT NULL Constraint wird verhindert, dass in den entsprechenden Spalten NULL-Werte enthalten sein dürfen. Nur möglich, wenn keine bestehenden Daten gegen NOT NULL verstossen.

```
ALTER TABLE tabellenname MODIFY spalte NOT NULL;
```

```
alter table dept modify dname not null;
```

Durch den NOT NULL Constraint wird erreicht, dass in der Spalte dname etwas eingetragen werden muss. Wird gegen diese Regel verstoßen, erscheint eine

```
insert into dept (deptno) values (60);
ORA-01400:
Einfügen von NULL in ("SCOTT"."DEPT"."DNAME") nicht möglich.
```

Durch die Insert-Anweisung wird versucht, einen Datensatz ohne Abteilungsname hinzuzufügen

1.5.6 Constraintangabe bei Tabellenerstellung

```
create table OfficialCar
  (
   CarNr int constraint PK1 PRIMARY KEY,
   CarName varchar2(50) constraint  nn NOT NULL,
   VIN int constraint UQ1 UNIQUE,
   color varchar2(20) constraint ck1 check (color in('Black','White')),
   deptno int constraint fk1 references scott.dept(deptno)
  )
;
```

1.5.7 CONSTRAINT Aktivierung

```
alter table OfficialCar enable constraint uq1;
```

1.5.8 CONSTRAINT Deaktivierung

```
alter table OfficialCar disable constraint uq1;
```

1.5.9 Übungen 08

Übungen siehe Seite 251.

1.6 Data Manipulation Language (DML)

1.6.1 INSERT

Normalerweise werden Datensätze mit Hilfe eines benutzerfreundlichen Front-End-Programms in die Tabellen eingetragen. Intern geschieht aber nichts weiter wie ein Aufruf der INSERT-Anweisung. Für das Erzeugen von Testdaten ist es ebenfalls recht hilfreich, die INSERT-Anweisung zu beherrschen. Eine INSERT-Anweisung wird erst durch ein **COMMIT** festgeschrieben. Alle Spalten,die auf NOT NULL definiert wurden, müssen selbstverständlich angegeben werden. Der allgemeine Syntax einer INSERT-Anweisung sieht folgendermaßen aus:

```
insert into tablename
         (spalte1,spalte2 ... spalte n)
  values (value1, value2  ... value n)
;
```

```
insert into emp
               (empno,ename,deptno,job)
        values (11,'Hager', 20,'SALESMAN')
;
```

Es wird in die Tabelle Mitarbeiter (emp) ein neuer Datensatz hinzugefügt. Es handelt sich um den neuen Mitarbeiter Herrn Hager mit der Mitarbeiternummer 11, der Abteilungsnummer 20 und dem Job SALESMAN.

```
insert into emp
  values (10,'Huber','SALESMAN',7902,'1.1.1990',2000,NULL,10)
;
```

Da hier hinter emp keine Spalten angegeben wurden, müssen alle Spalten in der entsprechenden Reihenfolge im Teil VALUES angegeben werden. Erzeugt wird durch die Anweisung ein neuer Mitarbeiter mit der Mitarbeiternummer 10, dem Namen Huber, dem Job SALESMAN, dem Vorgesetzten mit der Nummer 7902 (Herr Ford), dem Einstellungsdatum 1.1.1990, dem Gehalt 2000, ohne Kommission und der Abteilung Nummer 10.

Datensätze können auch auf Basis einer Abfrage hinzugefügt werden. Mit dieser Methode ist es zum Beispiel möglich, sehr schnell sehr große Datenmengen zu Testzwecken zu erzeugen. Denn was nutzt Ihnen eine Testdatenbank, die nur ein paar Tausend Datensätze besitzt. Als erstes erzeugen wir eine Tabelle namens SALESMAN mit zwei Spalten namens empno und ename.

```
create table salesman(empno int, ename char(20));
```

Anschließend wird diese Tabelle mit einer INSERT- Anweisung und einer Unterabfrage mit Daten gefüllt.

```
insert into salesman
  select empno,ename from emp where job = ,'SALESMAN'
;
select * from salesman;
EMPNO ENAME
---------------
```

```
7499  ALLEN
7521  WARD
7654  MARTIN
7844  TURNER
  10  Huber
  11  Hager
```

Mit Hilfe von Skripten können Sie ebenfalls Daten in eine Tabelle einfügen. Schauen wir uns folgendes kleine Skript etwas genauer an (eine detaillierte Einführung in Skripte würde den Rahmen dieser Unterlage sprengen).

```
ACCEPT Mitarbeiternummer PROMPT 'Geben Sie die Mitarbeiternummer an'
ACCEPT Mitarbeitername PROMPT 'Geben Sie den Mitarbeiternamen an'
ACCEPT Job PROMPT 'Geben Sie den Job des Mitarbeiters an'
INSERT INTO emp(empno,ename,job)
         VALUES(&mitarbeiternummer,'&mitarbeitername','&job')
;
```

Speichern Sie dieses Skript ab (z.B. als `~/test/skript1.sql`) und führen Sie es innerhalb von SQL PLUS aus (`start ~/test/skript1.sql`). Sie werden nacheinander zur Eingabe der entsprechenden Werte aufgefordert und am Ende wird der Datensatz eingefügt.

1.6.2 UPDATE

Mit der Anweisung UPDATE ist es möglich, Datensätze innerhalb einer Tabelle zu aktualisieren bzw. zu ändern. Eine Update-Anweisung kann durchaus mehrere Datensätze 'zugleich' ändern. Mit der WHERE-Einschränkung können Sie angeben, welche Datensätze geändert werden sollen.

```
update tablename set spalte = neuen_spaltenwert where bedingung;
```

```
UPDATE EMP SET sal = sal*1.1 where job = 'SALESMAN';
```

Das Gehalt aller SALESMAN wird um 10% erhöht.

```
UPDATE EMP SET job = 'Verkäufer' where job= 'SALESMAN';
```

```
select ename,job from emp;
```

```
ENAME       JOB
-----------------------
SMITH       CLERK
ALLEN       Verkäufer
WARD        Verkäufer
JONES       MANAGER
MARTIN      Verkäufer
BLAKE       MANAGER
CLARK       MANAGER
SCOTT       ANALYST
KING        PRESIDENT
TURNER      Verkäufer
ADAMS       CLERK
JAMES       CLERK
FORD        ANALYST
MILLER      CLERK
```

Der Job SALESMAN wird in der gesamten Tabelle emp umbenannt in Verkäufer.

1.6.3 DELETE

Löschoperationen werden in SQL durch die DELETE-Anweisung ausgeführt. Auch Löschoperationen müssen genau wie auch INSERT- und UPDATE-Anweisungen durch ein **COMMIT** bestätigt werden. Falls die Löschoperationen am Ende doch nicht durchgeführt werden sollen, so können diese mit **ROLLBACK** zurückgerollt werden.

```
delete from tabellenname where bedingung
```

```
DELETE FROM emp where job = 'SALESMAN';
```

Es werden alle Datensätze der Tabelle Mitarbeiter (emp) gelöscht,bei denen der Job SALESMAN ist.

```
DELETE FROM emp;
```

Es werden alle Datensätze der Tabelle Mitarbeiter (emp) gelöscht.

1.7 Transaction Control

1.7.1 ROLLBACK und COMMIT

Ziel einer Datenbank ist es, die Datenbank von einem konsistenten Zustand in den nächsten konsistenten Zustand zu überführen.

Stellen Sie sich vor, Sie erhöhen den Preis von 1000000 Produkte um 3 %. Nachdem 700000 Produkte aktualisiert wurden, gibt es einen Fehler. Die Datenbank befindet sich nicht mehr in einem konsistenen Zustand. Der erste konsistente Zustand sind die Produkte vor der Erhöhung und der nächste konsistente Zustand wäre der Zustand nach der Änderung aller 1000000 Produkte.

Führen Sie unter Oracle Einfüge-, Änderungs- oder Löschaktionen aus, so werden die beteiligten Datensätze noch nicht festgeschrieben. Erst durch ein COMMIT werden die geänderten Datensätze wirklich festgeschrieben. Analog zum Festschreiben per Commit gibt es auch die Möglichkeit, die durchgeführten Änderungen rückgängig zu machen. Die Anweisung hierfür lautet ROLLBACK. Wir wollen nun die beiden Anweisungen COMMIT und ROLLBACK in einem kleinen Beispieletwas genauer betrachten:

```
insert into dept (deptno,dname) values (50,'MARKETING');
insert into dept (deptno,dname) values (60,'MERCHAND');
commit;
```

In die Tabelle Abteilung dept werden zwei neue Datensätze hinzugefügt. Diese Datensätze werden mit COMMIT festgeschrieben.

```
insert into dept (deptno,dname) values (70,'DEP1');
insert into dept (deptno,dname) values (80,'DEP2');
rollback;
```

In die Tabelle Abteilung dept werden zwei neue Datensätze hinzugefügt. Diese Datensätze zum Abschluss wieder zurückgerollt. Das bedeutet, die beiden Datensätze sind nicht eingefügt worden.

1.7.2 Automatisches COMMIT

Bei einigen Befehlen ist ein automatisches COMMIT eingebaut. Das bedeutet, wenn Sie Ihre Datenbank geändert, aber die Änderungen noch nicht mit COMMIT permanent gemacht haben und dan einen dieser Befehle ausführen, werden die Änderungen automatisch dauerhaft gespeichert. Die gebräuchlichsten Befehle mit einem automatischen COMMIT sind:
ALTER, CREATE, DROP und RENAME.

1.7.3 SAVEPOINT

Das setzen von Savepoints erlaubt interime Labels innerhalb einer Transaktion zu setzen und diese explizit bei einem ROLLBACK anzusprechen. Auch wenn SAVEPOINTS gesetzt sind ist immer ein Gesamt ROLLBACK möglich. Das explizite ROLLBACK zu einem SAVEPOINT macht alle Datenmanipulationen bis zurück zum angegebenen SAVEPOINT rückgängig. LOCKS, welche nach dem SAVEPOINT gesetzt wurden, werden freigegeben. Änderungen vor dem

SAVEPOINT können nach einem ROLLBACK TO SAVEPOINT mit COMMIT festgeschrieben bzw. mit ROLLBACK rückgängig gemacht werden.

SAVEPOINTS sind Befehls- und nicht Transaktions-orientiert. Sie werden in Error-Handlers verwendet, um geleistete Arbeit zu schützen. Oracle setzt vor jedem Befehl einen impliziten SAVEPOINT. Dieser ist allerdings nicht explizit (vom User) ansprechbar.

Beispiel: SAVEPOINT / ROLLBACK TO

```
INSERT INTO personal (name,ort) values ('Meier','Luzern');
SAVEPOINT meier;
INSERT INTO personal (name,ort) values ('Huber','Luzern');
SAVEPOINT huber;
INSERT INTO personal (name,ort) values ('Weber','Luzern');
SAVEPOINT weber;
ROLLBACK TO meier;
-- Macht eine ROLLBACK bis zum SAVEPOINT meier, der erste INSERT bleibt also erhalten.
COMMIT;
--Schreibt das erste INSERT fest.
```

1.7.4 Lesekonsistenz

Nehmen wir einmal an, dass Scott die Tabelle Mitarbeiter (emp) ändert. Er fügt zwei neue Datensätze hinzu.

```
insert into emp(empno,ename,job) values(21,'Kluge','SALESMAN');
Insert into emp(empno,ename,job) values(22,'Hinze','SALESMAN');
```

```
Select empno,ename,job from emp where empno between 21 and 22;
EMPNO  ENAME       JOB
 ----------  ----------  --------------------
         21  Kluge       SALESMAN
         22  Hinze       SALESMAN
```

Die eingefügten Datensätze sind für Scott in seiner Sitzung bereits zu sehen. Und zwar noch bevor er ein COMMIT durchgeführt hat. Ein anderer Benutzer ist ebenfalls am Oracle-Server angemeldet. Auch dieser Benutzer fragt die Tabelle Mitarbeiter (emp) ab:

```
select empno,ename,job from scott.emp where empno between 21 and 22;
```

Es wurden keine Zeilen ausgewählt Die von scott geänderten Daten sind noch nicht verfügbar. Erst, nachdem Scott seine Änderungen mit einem Commit bestätigt, sind diese Daten auch für alle anderen am Oracle-Server angemeldeten Benutzer zu sehen. Diese Vorgehensweise wird als Lesekonsistenz bezeichnet. Datenänderungen sind nur in der Sitzung zu sehen, in der die Änderungen durchgeführt wurden.

1.7.5 Übungen 07

Übungen siehe Seite 250.

1.8 Data Control Language (DCL), CREATE/DROP USER/ROLE

1.8.1 CREATE USER

```
CREATE USER Benutzername
  IDENTIFIED BY Kennwort
  [DEFAULT TABLESPACE Standardtablespace]
  [TEMPORARY TABLESPACE temporärer Tablespace]
  [QUOTA 10M ON Kontingent für Tablespace]
;
```

```
CREATE USER bertab
  IDENTIFIED BY bertab
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp
  QUOTA 10M ON users
;
```

Es werden dabei aber keine Privilegien zugewiesen.

1.8.2 ALTER USER

```
ALTER USER user IDENTIFIED BY password;
```

```
ALTER USER scott IDENTIFIED BY lion;
```

1.8.3 DROP USER

```
DROP USER Benutzername;
```

```
DROP USER bertab;
```

1.8.4 CREATE ROLE

```
CREATE ROLE manager;
```

Die Rolle manager wurde erzeugt. Sie muss noch mittels GRANT mit Rechten versehen werden.

1.8.5 GRANT ... TO

GRANT ist ein DCL-Befehl. Wie bei allen DCL-Befehlen erfolgt bei Ausführung ein automatisches COMMIT.

Rechtevergabe an einen user:

```
GRANT privileges TO user;
```

```
GRANT connect TO bertab;
```

Rechtevergabe an eine Rolle:

```
GRANT privileges TO role;
```

```
GRANT create table, create view TO manager;
```

Es wurden die Rechte create table und create view der Rolle manager zugewiesen. In der Syntax erscheint nicht das Wort ROLE. Nun kann die Rolle einem User zugewiesen werden:

```
GRANT manager TO scott;
```

1.8.6 GRANT ... ON ... TO

Wichtige Berechtigungen: INSERT, UPDATE, DELETE und SELECT.

```
GRANT object_priv ON object TO user|role|PUBLIC [WITH GRANT OPTION];
```

PUBLIC erlaubt den Zugriff für alle User.
WITH GRANT OPTION Erlaubt die Weitervergabe von Rechten zu anderen Usern und Rollen.

(Dies kann aber bei Objektrechten nicht weitergereicht werden. ??)

```
GRANT insert ON employees TO bertab;
```

Der User bertrab darf nun INSERT in der Tabelle employees anwenden.

1.8.7 REVOKE ... ON ... FROM

```
REVOKE Berechtigung ON Tabellenname FROM Benutzer;
```

```
REVOKE insert ON employees FROM bertab;
```

1.9 Einführung in die PL/SQL-Programmierung

Grundsätzlich kann man einen PL/SQL-Block in drei Abschnitte unterteilen.

1. Deklarationsabschnitt
   Als erstes erfolgt die Deklaration von Variablen, Konstanten etc.

2. Ausführbarer Abschnitt
   Als zweites werden die entsprechenden Befehle in den ausführbaren Abschnitt eingetragen.

3. Exceptionsabschnitt
   Als letztes haben Sie einen Abschnitt mit Exceptions. Hier wird eine Ausnahmebehandlung durchgeführt.

Wir wollen die drei Abschnitte anhand eines kleinen Beispiels verdeutlichen. Die Syntax der hier verwendeten Befehle und Kontrollstrukturen wird in den weiteren Kapiteln genauer beschrieben.

1.9.1 Anonymer Block

```
-- Deklarationsabschnitt
declare
   v_zahl1 number(5) := 5;
   v_zahl2 number(5) DEFAULT 5;

-- Ausführbarer Abschnitt & Exception
begin
   select count(*) into anzahl from scott.emp;
   dbms_output.put_line(anzahl);
   -- Exception-Abschnitt
   Exception
     when too_many_rows then
     dbms_output.put_line('Sie haben zu viele Werte.');
       when others then
     dbms_output.put_line('unbekannte Ausnahmeverletzung.');
end;
```

Nach begin folgt der ausführbare Teil des Blockes. Hier wird zuerst die Ausgabe per dbms_output aktiviert.

Es gibt viele Datensätze in der Tabelle scott.emp. Somit wird die Exception **too_many_rows** aufgerufen. Diese Exception wird immer dann aufgerufen, wenn mit Hilfe einer SELECT-Anweisung mehr als ein Wert in eine Variable geladen werden soll. Nach dieser Exception wird der Block beendet. Es gibt neben der Exception too_many_rows natürlich noch weitere Exceptions, die später behandelt werden.

Sie können gern dieses kleine Programm testen. Speichern Sie die Datei unter beispielsweise ~/test/t1.sql ab und rufen Sie dieses Programm unter sqlplus mit dem Befehl `start ~/test/t1.sql` auf.

1.9.2 Variablendeklaration

Neben den üblichen Variablen gibt es bei PL/SQL noch weitere Datentypen, die beim Erstellen eines Blockes benutzt werden können.

Boolean	Boolsche Variable mit den Zuständen TRUE oder FALSE.
Binary_integer	Zahlen von -2147483647 bis +2147483647.
natural	Zahlen von 0 bis +2147483647.
positive	Zahlen von 1 bis +2147483647.
%Type	Zuweisung von Datentypen äquivalent zum Datentype einer Spalte.
%rowtype	Zusammengesetzter Datentyp, äquivalent zu einer Tabellenzeile.

Deklaration einer Variable mit %type

Mit dem Datentype %type können Sie einer Variablen den gleichen Datentyp zuweisen, den eine bestehende Tabellenspalte besitzt. Dies ist hilfreich, wenn bestimmte Variablen mit Werten aus einer Tabelle versehen werden sollen. Sie müssen sich in diesem Fall keine Gedanken mehr um den Datentyp dieser Variablen machen. Sie nehmen einfach für diese Variable den Datentyp, den die Spalte, deren Wert in die Variable geladen werden soll, besitzt.

```
Variable Tabellenname.Spalte\%type
```

Deklaration einer Variable mit %rowtype

Mit dem Datentype %rowtype können Sie in einer Variablen alle Spalten einer Tabelle speichern und diese anschließend abrufen. Die entsprechenden Spalteninhalte sind mit Variablenname.Spaltenname abrufbar.

```
Variable Tabellenname\%rowtype
```

1.9.3 IF ... THEN ... ELSE

Oracle PL/SQL bietet die Möglichkeit, Bedingungen zu formulieren und in Abhängigkeit des Wahrheitsgehaltes dieser Bedingungen einen Block auszuführen oder nicht auszuführen.

Folgendes Beispiel soll den Aufbau einer IF THEN ELSE-Anweisung darstellen: Durch IF anzahl <= 10 then ... wird überprüft, ob die Variable anzahl kleiner oder gleich 10 ist. Wenn diese Bedingung wahr ist, so wird der Block ausgeführt, der sich dem THEN befindet. In unserem Beispiel ist das die Anweisung dbms_output.put_line('Wenig'). Diese Anweisung gibt lediglich die Zeichenkette Wenig auf dem Bildschirm aus.

Ist die Bedingung 'anzahl kleiner oder gleich 10' falsch, so geht Oracle zum nächsten ELSIF (Achtung: Nicht ELSEIF!!). Durch ELSIF anzahl <= 20 then ... wird überprüft, ob die Variable anzahl kleiner oder gleich 20 ist. Wenn diese Bedingung wahr ist, so wird der Block ausgeführt, der sich dem THEN befindet.

In unserem Beispiel ist das die Anweisung dbms_output.put_line('Mittel'). Diese Anweisung gibt lediglich die Zeichenkette Mittel auf dem Bildschirm aus. Ist die Bedingung 'anzahl kleiner oder gleich 20' falsch, so führt Oracle den Block aus, der sich nach dem ELSE befindet. In unserem Falle also dbms_output.put_line('viel')

```
IF Bedingung THEN
  Anweisungen;
ELSIF Bedinung THEN
  Anweisungen;
ELSIF Bedinung THEN
  Anweisungen;
  ...
ELSE
  Anweisungen;
END IF;
```

```
If anzahl < 100 then
dbms_output.put_line('Wenig');
Elsif anzahl between 100 and 200 then
      dbms_output.put_line('Mittel');
Else
  dbms_output.put_line('Viel');
End If;
```

1.9.4 LOOP ... END LOOP

Eine LOOP ... END LOOP-Schleife ohne EXIT-Kriterium ist eine Endlosschleife. Die Anweisungen werden unendlich wiederholt. Da Endlosschleifen in der Praxis eher selten benötigt werden, wird ein EXIT-Kriterium eingesetzt.

```
Loop
  exit when Bedingung;
End Loop;
```

Im folgenden Beispiel wird als EXIT-Kriterium keine IF-Bedingung benutzt, sondern eine EXIT WHEN-Anweisung. Wenn die Variable Zaehler größer als 10 ist, so wird die LOOP ... END LOOP-Schleife verlassen. Es wird zur Anweisung hinter dem END LOOP gesprungen und dort wird ganz normal fortgefahren.

```
-- Vorher muss natürlich a deklariert worden sein.
Loop
       exit when a=10;
       a:=a+1;
       dbms_output.put_line(a);
End Loop;
```

1.9.5 WHILE LOOP ... END LOOP

Anders als bei der normalen LOOP ... END LOOP-Anweisung ist das EXIT-Kriterium bei der WHILE LOOP ... END LOOP-Anweisung bereits über das WHILE integriert. Die Schleife wird so lange durchlaufen, wie die Variable Zaehler kleiner oder gleich 10 ist. So bald die Variable Zaehler größer als 10 ist, wird die Schleife beendet und es wird mit der Anweisung hinter END LOOP fortgefahren.

```
WHILE Bedingung LOOP
  Anweisungen;
END LOOP;
```

```
-- Vorher muss natürlich a deklariert worden sein:
While a<>11 Loop
   dbms_output.put_line(a);
   a:=a+1;
End Loop;
```

1.9.6 FOR ... IN .. LOOP ... END LOOP

Durch eine FOR LOOP ... END LOOP-Schleife wird ein Block n-mal durchlaufen. Die Anzahl der Durchläufe wird durch die Grenzwerte hinter IN definiert. Im folgenden Beispiel wird die Schleife demnach 11 mal durchlaufen. Begonnen wird bei 0, anschließend wird hochgezählt und wenn die Variable zaehler den Wert 10 erreicht hat, wird die Schleife beendet.

```
FOR Zählvariable IN untere_Grenze .. obere_Grenze LOOP
  Anweisungen;
END LOOP;
```

```
-- x muss nicht deklariert werden:
For x In 1 .. 10 Loop
    dbms_output.put_line(x);
End Loop;
```

1.9.7 Erstellen einer Prozedur

```
CREATE PROCEDURE name [ ( parameter [, parameter ] ) ]
IS
   -- Deklarationsteil
   BEGIN
   -- Programmteil
   EXCEPTION
   -- Ausnahmebehandlung
   END;
;
```

```
set serveroutput on;
create or replace procedure p2(zae_beg int,zae_end int)
is
anzahl number(5) := 1;
a int:=1;
  begin
    for x in zae_beg .. zae_end loop
      select count(*) into anzahl from sys.products where supplierid=x;
      dbms_output.put_line(anzahl);
    end loop;
  Exception
    when too_many_rows then
      dbms_output.put_line('Sie haben zu viele Werte');
      when others then
        dbms_output.put_line('unbekannte Ausnahmeverletzung');
  end;
/
show errors;

-- Aufrufen mit exec p2(1,10)
```

1.9.8 Tipps und Tricks

Dynamisches Erstellen von SQL-Anweisungen (Für DDL etc.)

```
EXECUTE IMMEDIATE 'CREATE TABLE X(A DATE)';
```

Fehlerbehandlung

```
begin
   select bezeichnung into v_bezeichnung
    from einheit
    where einheit_kurz = 'm';
   if v_bezeichnung <> 'Meter' then
    raise EINHEIT_FEHLER;
   end if;
exception when EINHEIT_FEHLER then
   ...
end;
```

1.9.9 Beispiele

Erstellen eines Cursors (Beispiel 1)

```
DECLARE
emp_id CHAR(9);
FNAME VARCHAR(12);
LNAME VARCHAR(20);
CURSOR CUR1 IS
SELECT employeeid, firstname, lastname
FROM employees ORDER BY employeeid;
BEGIN
    OPEN CUR1;
    FETCH CUR1 INTO emp_id, FNAME, LNAME;
    WHILE (CUR1%FOUND) LOOP
    FETCH CUR1 INTO emp_id, FNAME, LNAME;
    END LOOP;
CLOSE CUR1;
END;
```

Erstellen eines Cursors (Beispiel 2)

```
SET SERVEROUTPUT ON
DECLARE CURSOR video_cursor
IS
  SELECT title, mpaa_rating FROM video
    WHERE hit_status = 1
    ORDER BY title;
  BEGIN
    FOR video_rec
      IN video_cursor
      LOOP
        DBMS_OUTPUT.PUT_LINE (video_rec.title || ',' || video_rec.mpaa_rating);
      END LOOP;
END;
/
```

1.9.10 Übungen 09

Übungen siehe Seite 253.

1.10 Neuerungen in Oracle 9i

1.10.1 NULLIF()

Diese neue Funktion vergleicht zwei Werte oder Ausdrücke auf Gleichheit. Sind beide gleich, so wird NULL zurückgegeben, sonst der erste Wert.

```
SELECT ename, NULLIF(deptno, 20) FROM SCOTT.emp;
```

1.10.2 CASE

```
SELECT
  CASE WHEN deptno = 20
    THEN NULL
  ELSE
    deptno
  END
  FROM SCOTT.emp
;
```

1.10.3 COALESCE()

COALESCE ist eine generalisiertere Form von NVL. An COALESCE kann im Gegensatz zu NVL eine beliebig lange Liste von Argumenten übergeben werden, Minimum sind zwei Argumente. COALESCE gibt aus der Liste den ersten Wert zurück, der nicht NULL ist.

```
SELECT COALESCE(comm, sal) FROM SCOTT.emp;
```

1.10.4 JOIN

Die JOIN-Befehle sind FROM untergeordnet.

CROSS JOIN

Es wird ein Kreuzprodukt erstellt. Ein ON kommt in der Syntax nicht vor.

```
select e1.ename,e2.ename
  from scott.emp e1
    cross join scott.emp e2;
```

NATURAL JOIN

Alle gleichnamige Spalten mehrerer Tabellen werden automatisch verknüpft. Ein ON kommt in der Syntax nicht vor.

```
SELECT deptno, ename, dname
  FROM emp
    NATURAL JOIN dept;
```

USING JOIN

Beim USING JOIN wird der gemeinsame Spaltenname explizit angegeben. Dies ist bei mehreren, gleichnamigen Spalten notwendig. Ein ON kommt in der Syntax nicht vor.

```
SELECT deptno, ename, dname
  FROM emp
    JOIN dept USING (deptno)
;
```

LEFT OUTER JOIN

```
select d.deptno,e.ename
  from scott.dept d
    left outer join scott.emp e
      on e.deptno = d.deptno
;
```

RIGHT OUTER JOIN

```
select d.deptno,e.ename
  from scott.emp e
    right outer join scott.dept d
      on e.deptno = d.deptno
;
```

FULL OUTER JOIN

```
select d.deptno,e.ename
  from scott.emp e
    full outer join scott.dept d
      on e.deptno = d.deptno
;
```

INNER JOIN

```
select d.deptno,e.ename
  from scott.dept d
    join scott.emp e
      on e.deptno = d.deptno
;
```

1.10.5 MERGE – INSERT + UPDATE

Ab Version 9.0 bietet Oracle die Möglichkeit, INSERT- und UPDATE-Statements zu kombinieren. Werden beispielsweise ständig neue Daten in eine Hilfstabelle eingespielt, anhand derer Einträge in Tabellen zum Teil ergänzt (bei komplett neuen Datensätzen), zum Teil aber auch geändert werden müssen (bei bereits vorhandenen Datensätzen), so kann dies mit MERGE in einem Schritt erfolgen.

```
CREATE TABLE annsal(empno NUMBER, peryear NUMBER);
INSERT INTO annsal (empno) SELECT empno FROM emp WHERE deptno = 10;

SELECT * FROM annsal;
EMPNO PERYEAR
7782
7839
7934

MERGE INTO annsal a
  USING emp e
  ON (a.empno = e.empno)
  WHEN MATCHED THEN
    UPDATE SET a.peryear = (e.sal *12 + NVL(e.comm,0))
  WHEN NOT MATCHED THEN
    INSERT (a.empno, a.peryear) VALUES (e.empno, e.sal *12 + NVL(e.comm,0))
;

SELECT * FROM annsal;
EMPNO  PERYEAR
 7782    29400
 7839    60000
 7934    15600
 7902    36000
 7900    11400
 7521    15500
 7654    16400
 7566    35700
 7844    18000
 7499    19500
 7698    34200
 7788    36000
 7369     9600
 7876    13200
```

1.10.6 DEFAULT

Ab Version 9i kann nunmehr das Wort DEFAULT auch in Einfüge- und Aktualisierungsoperationen genutzt werden.

Umgang mit DEFAULT

```
create table test(Nummer int,city varchar2(100) default 'Berlin');
```

Einfügen von Datensätzen

```
insert into test(nummer,city) values(1,default);
```

Aktualisieren von Datensätzen

```
update test set city = default where nummer = 1;
```

1.10.7 Übungen 10

Übungen siehe Seite 255.

Kapitel 2

Fundamentals

2.1 Oracle-Server Bestandteile

Oracle Server: DB-Instance + Datenbank

- Instance
  - System Global Area (SGA)
    - Shared Pool
      - Library Cache
      - Data Dictionary Cache
    - Database Buffer Cache
    - Redo Log Buffer
    - Java Pool
    - Large Pool
  - Background Process Structures
    - PMON
    - SMON
    - DBWR
    - LGWR
    - CKPT
    - Optionale Background-Prozesse
      - ARCn
- Datenbank
  - Database Files (Operating System Files)
    - Datafiles
    - Control Files
    - Redo Logfiles
- Other Key Files
  - Parameter Files
  - Password File
  - Archives Redo Logfiles
- User und Server Prozesse
  - Server Process – Program Global Area (PGA)
  - User Process
- Andere Prozesse
  - Advances Queuing
  - Real Application Clusters
  - Shared Server
  - Advanced Replication
  - ...

2.1.1 Instance

Man startet nie eine Datenbank. Man startet eine Instanz. Diese ermöglicht den Zugriff auf die Datenbank. Eine Instanz öffnet eine und nur eine Datenbank.

Eine Instanz muss beim Start initialisiert werden. Dadurch werden die Initialisierungsparameter festgelegt. Dies erfolgt durch ein SPFILE bzw. ein PFILE (siehe Parameter Files, Seite 103).

System Global Area (SGA)

Beim Start einer Instanz wird Arbeitsspeicher für den SGA zugewiesen. SGA ist dynamisch anpassbar (ausser Gesamtgröße). Das heißt die Instanz braucht nicht heruntergefahren zu werden um die SGA-Konfiguration zu ändern. SGA_MAX_SIZE definiert die maximale Größe der SGA.

```
     DB_CACHE_SIZE (BUFFER_CACHE)
+ SHARED_POOL_SIZE
+   JAVA_POOL_SIZE
+  LARGE_POOL_SIZE
-----------------
<=    SGA_MAX_SIZE
```

Anzeige der Speicherzuweisung:

```
show sga;
```

Die Speicherzuweisung erfolgt schrittweise (Granulate). Die Granulat-Größe ist abhängig on der SGA-Größe (4MB bei SGA <128MB, ansonsten 16MB).

```
ALTER SYSTEM SET SGA_MAX_SIZE = xxxM;
```

Shared Pool: Der Share Pool speichert die zuletzt verwendeten SQL Statements und Datendefinitionen. SHARED_POOL_SIZE definiert die Größe des Buffers für shared SQL und PL/SQL (Default 16MB bei 64bit bzw. 64MB bei 64 bit).

Library Cache: Der Library Cache speichert die zuletzt verwendeten PL/SQL-Befehle mittels LRU-Algorithmus (Last Recentliy Used).

Data Dictionary Cache: Der Data Dictionary Cache speichert die zuletzt verwendeten Objekte (Tabellen, Views, ...).

Database Buffer Cache: Der Database Buffer Cache speichert die zuletzt verwendeten Kopien von Datenblöcken mittels LRU-Algorithmus. Der Database Buffer Cache besteht aus den Sub-Caches:
DB_CACHE_SIZE definiert die Größe des Cache für Standard-Blocks (Default 48MB bei UNIX bzw. 52 MB bei NT).
DB_KEEP_CACHE_SIZE ist die Größe des Keep Buffer Cache, welcher Blöcke zum Wiederverwenden speichert.
DB_RECYCLE_CACHE_SIZE ist die Größe des Recycle Cache, welcher nicht verwendete Blöcke aus dem Speicher entfernt.
DB_BLOCK_SIZE bestimmt die Primary Block Size.

Redo Log Buffer: Der Redo Log Buffer ist ein Circular Buffer und speichert Änderungen von Datenblöcken. LOG_BUFFER definiert die Größe des Redo Log Buffer in Bytes.

Java Pool: Der Java Pool ist optional und wird für Java verwendet. JAVA_POOL_SIZE definiert die Größe des Java Pools (Default 24MB).

Large Pool: Der Large Pool ist ein optionaler Speicherbereich im SGA. Er wird für Session Memory verwendet bei Shared Server. Weitere Anwendungen sind I/O, Backup/Restore (RMAN), ... Der Large Pool verwendet kein LRU-Algorithmus. LARGE_POOL_SIZE definiert die Größe des Large Pools (Default 0). LARGE_POOL_SIZE ist nicht dynamisch.

Background Process Structures

Beim Start einer Instanz werden die Background-Prozesse gestartet.

PMON: Fehlerhafte Prozesse im PGA werden durch den Process Monitor (PMON) aufgeräumt mittels:

- Zurückrollen der Transaktion
- Freigeben von Locks
- Freigeben anderer Resourcen.
- Neustart von toten Dispatchern.

SMON: Bei einem Crash können keine Informationen des SGA auf die Festplatten geschrieben werden. In diesem Fall führt der System Monitor (SMON) ein automatisches Instanz-Recovery durch. Dies beinhaltet folgende Schritte:

1. Daten, die nicht vom Database Buffer Cache in die Database Files geschrieben wurden, werden aus den Redo Logs gelesen und in die Database Files geschrieben.
2. Öffnen der Datenbank, so das User sich einloggen können. Daten, die nicht durch abgebrochene Transaktionen gespert sind (Locks), stehen zur Verfügung.
3. Nicht commitete Transaktionen werden zurückgerollt.

Der SMON führt auch Speicherverwaltungsaufgaben durch.

- Benachbarte Bereiche in den Database Files werden zusammengefasst.
- Temporäre Segmente werden in den Database Files freigegeben.

DBWRn: Der Database Writer schreibt Dirty Buffers von dem Database Buffer Cache in die Database Files. Er verzögert dies aber bis eines der folgenden Ereignisse eintritt:

- Ein Checkpoint wird gesetzt.
- Der Bedarf an Dirty Buffer übersteigt einen Grenzwert.
- Timeout
- Ein Real Application Cluster (RAC) Ping Request erfolgte.
- Ein Tablespace wird offline gesetzt.
- Ein Tablespace wird read only gesetzt.
- Eine Tabelle wird mit DROP oder TRUNCATE gelöscht.
- `ALTER TABLESPACE tablespace name BEGIN BACKUP;`

LGWR: Der Log Writer (LGWR) schreibt sequentiel vom Redo Log Buffer in die Redo Log Files wenn eines der folgenden Ereignisse eintritt:

- COMMIT
- Der Redo Log Buffer 1/3 voll ist.
- Wenn mehr als 1MB an Änderungen im Redo Log Buffer sind.
- Alle drei Sekunden.
- Bevor der DBWR schreibt.

Da Redo Files für das Recovery benötigt werden, bestätigt der LGWR einen COMMIT-Befehl erst nach dem erfolgreichen Schreiben in die Redo Logs. In den Redo Logs wird jeweils der gesamte Datensatz geschrieben. Der LGWR kann den DBWR beeinflussen, dass dieser in die Datafiles schreibt.

CKPT: Alle drei Sekunden schreibt der CKPT-Prozess Daten in die Control-Files, um die Stellen in den Redo Logs zu kennzeichnen, bei denen ein Recovery beginnen muss. Diese Daten nennt man Checkpoints. Erfolgt ein Log Switch wird auch ein Checkpoint geschrieben.

Checkpoints werden aus folgenden Gründen iniziiert:

- Um sicherzustellen, das die modifizierten Datenblocks regelmäßig auf die Fetplatten geschrieben werden.
- Zur Verringerung des Zeitaufwandes beim Instanz-Recovery.
- Zur Sicherstellung, das bei einem Shutdown alle committeten Daten in die Database Files geschrieben werden.

Beachte: Der CKPT schreibt keine Datanblocks auf die Festplatten oder Redo-Blocks in die Redo-Logs.

Optionale Background-Prozesse

ARCn: Von den optionalen Background-Prozessen wird hier nur der ARCn besprochen. Der ARCn dient zum Erstellen der Archive Log-Files. Die Achivie Log dienen dem Offline-Recovery.

Die Online Redo Logs werden nacheinander geschrieben. Wenn es drei Redo Log Files gibt, wird erst in die erste Datei geschrieben. Ist diese voll, wird in die zweite Datei geschrieben. Ist das zweite Redo Log File voll, wird in die dritte Redo Log Datei geschrieben. Ist die dritte Datei voll, beginnen die Schreiboperationen wieder bei der ersten Datei. Die ursprünglichen Informationen der ersten Datei gehen dabei verloren. Um dies zu verhindern, gibt es die Möglichkeit vollgeschriebene Redo Log Files zu archivieren. Man nennt diese dann Offline Redo Logs bzw. Archive Logs und den Prozess der dies steuert den Archiver (ARCn). Die Offline Redo Logs sollten auf eine separaten Platte gespeichert werden und von dort aus gesichert werden.

Die Voreinstellung ist NOARCHIVELOG, dass heisst es werden keine Archive Logs geschrieben. Dies sollte unbedingt auf den ARCHIVELOG Mode geändert werden.

2.1.2 Datenbank

Database Files – Die physikalische Struktur

Die Datenbank Dateien, auch Operating System Files genannt, speichern physikalisch die Datenbank-Informationen. Sie stellen auch ein Recovery bei Fehlern sicher.

Datafiles: Die Datafiles enthalten die aktuellen Daten der Datenbank.

Control Files: Die Control Files ermöglichen die Überprüfung und Sicherstellung der Datenbank-Integrität.

Redo Logfiles: Die Redo Logfiles enthalten Informationen über die letzten Änderungen und dienen zur Online-Sicherung bzw. -Recovery. Beim Online Recovery braucht die Datenbank nicht heruntergefahren werden. Redo Logfiles sind keine Log Files im herkömmlichen Sinn. Sie sind Binärdateien.

Logische Struktur einer Datenbank

Eine Datenbank enthält mindestens einen Tablespace. Ein Tablespace enthält einen oder mehrere Segmente. Ein Segment besteht aus Extents. Extents bestehen aus Blöcken. Blöcke sind die kleinste Einheit für Lese- und Schreiboperationen.

```
Datenbank
  Tablespace
    Segment
      Extent
        Block
```

Die Standard-Blockgröße ist 8k (DB_BLOCK_SIZE). Diese sollte der Betriebssystem-Blockgröße entsprechen. Die Blockgröße wird bei der DB-Erstellung festgelegt und kann später nicht geändert werden.

2.1.3 Other Key Files

Parameter Files – (S)PFile: Die Parameterfiles initialisieren die Instanz beim Start (siehe Seite 103).

Password File: Mit Hilfe des Password Files erfolgt die Authentifizierung von Usern.

Archives Redo Log Files: Archives Redo Log Files sind archivierte Redo Log Dateien. Sie werden nicht mehr beschrieben und können gesichert werden. Sie dienen dem Offline-Recovery.

2.1.4 User- und Server Prozesse

Die User- und Server-Prozesse ermöglichen das Ausführen von SQL-Statements. User-Prozesse nehmen nie direkten Kontakt zur Instanz auf. Dazu wird ein Server-Prozess gestartet. Für diese Zeit wird also eine Session erzeugt.

Dedicated Server vs. Shared Server: Man unterscheidet eine eins-zu-eins Korrespondenz zwischen User- und Server-Prozess (Dedicated Server Connection) oder eine n-zu-eins Korrespondenz (Shared Server Connection).

Dedizierte Server-Konfiguration: Bei der dedizierten Server-Konfiguration ist jedem Benutzer-Prozess ein eigenen Server-Prozess zugeordnet. Wird eine neue Benutzer-Verbindung zur Datenbank hergestellt, so wird für diesen Benutzerprozess ein Server-Prozess zur Verfügung gestellt. Dieser Server-Prozess dient einzig und allein diesem einen Benutzer-Prozess. Falls der Benutzer-Prozess Freiraum hat, so ist auch der zugehörige Server-Prozess beschäftigungslos.

Shared Server: Die Shared Server Konfiguration wird seltener benutzt. Verwendung findet diese bei Internet-Datenbank-Konfigurationen. Siehe Seite 158.

Server Process – Program Global Area (PGA) Der Listener-Prozess registriert Verbindungswünsche von User-Prozessen. Server Prozesse werden dann auf Verlangen von User-Prozessen gestartet und führen stellvertretend dessen SQL-Statements aus und liefert die Ergebnisse an den User-Prozess. Der Server Process läuft auf der Maschine, auf der auch die Instanz läuft. Dabei wird für jeden User ein Dispatcher erstellt. Prioritäten kann man dabei nicht zuweisen. Wenn ein Server-Prozess startet, wird ihm Arbeitsspeicher zugewiesen. Diesen nennt man Program Global Area (PGA). Nach dem Beenden des User-Prozesses wird der PGA wieder freigegeben. Der PGA besteht aus diesen Komponenten:

- Private SQL Area
  - Persistent Area
  - Run-Time Area
- Session Memory
- SQL Work Areas

User Process Der User-Prozess läuft meist auf der lokalen Maschine des Benutzers bzw. auf einem Applikationsserver.

2.2 Getting Started with the Oracle Server

2.2.1 Administrator User, Rollen und Privilegien

Die User SYSTEM und SYS werden automatisch bei der Erstellung der Datenbank erstellt.

User SYS

Default-Passwort: change_on_install

Der User SYS ist Eigentümer des Data-Dictionary.

User SYSTEM

Default-Passwort: manager

Der User SYSTEM ist Eigentümer aller internen Tabellen und Views, die von den Oracle Tools verwendet werden.

Rolle DBA

Den Usern SYS und SYSTEM ist die DBA-Rolle zugewiesen.

Privilegien SYSDBA/SYSOPER

Spezielle Privilegien sind SYSDBA und SYSOPER. SYSOPER kann die Instanz starten, herunterfahren, (un-)mounten und bedingt recovern. SYSDBA hat die gleichen Rechte wie SYSOPER. Zusätzlich kann man auch mit diesem Privileg neue Datenbanken erzeugen und vollständiges Backup und Recovery durchführen. Als User SYS und SYSTEM sollte man entweder sich mit den Privilegien SYSDBA oder SYSOPER anmelden.

2.2.2 Oracle Enterprise Manager

Anmelden Repository

Im Repository befinden sich die Meta-Daten über einem OEM. Meta-Daten sind Daten über Daten. Es geht auch ohne Repository. Dann erfolgt halt die Verbindung über Standalone.

Anmelden Standalone

Bei der Anmeldung 'Standalone' wird sich ohne Verwendung eines Repositories verbunden.

2.3 Die Oracle Instance

Ermitteln der SID

```
select name, value
  from v$parameter
  where name = 'instance_name';
```

2.3.1 Initialisierungsparameter

Es gibt zwei Arten von Parametern.

- Explizit – Diese Werten sind in den Parameterfile gespeichert.
- Impizit – Diese Werte sind Default-Werte.

Anzeige der Initialisierungsparameter in der OEM-Konsole. OEM-Konsole starten.

```
oemapp console
```

Netzwerk — Datenbanken — SID — Instanz — Konfiguration
Alle Initialisierungsparameter

Beispiele

```
MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHlSTORY, MAXDATAFILES, MAXINSTANCES

UNDO_MANAGEMENT = AUTO
undo tablespace created with database creation, when undo_tablespace omitted

UNDO_TABLESPACE = undotbs
choosing/switching undo tablespace

CONTROL_FILES = path/filename
if not specified, control files created with database creation in OMF directory

DB_CREATE_FILE_DEST = path
datafiles, tempfiles OMF
(if db_create_online_log_dest_n not specified: also redo log files, control files)

DB_CREATE_ONLINE_LOG_DEST_1 ? 5 = path
redo log files, control files OMF

BACKGROUND_DUMP_DEST = path
location alert log file (and background process trace files)

LOG_CHECKPOINTS_TO_ALERT = TRUE
monitoring checkpoints in alert log file

LOG_ARCHIVE_START = TRUE
starting the archiver process

FAST_START_MTTR_TARGET = seconds
max time needed for instance recovery,
```

```
determines the number of buffers being written by DBWn

REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE
password authentication single instance

SORT_AREA_SIZE = bytes | k | m
uniform size of the temporary tablespace should be a multiple of the SORT_AREA_SIZE

RESOURCE_LIMIT = TRUE
enforce resource limits
```

2.3.2 Parameterfiles zur Initialisierung

Die Parameterfiles initialisieren die Instanz beim Start. Z.B. werden durch die Initialisierungsparameter die Größen der Speicherstrukturen im SGA festgelegt. Dies erfolgt bis 8i durch das PFILE. Ab 9i erfolgt die Initialisierung standardmäßig per SPFILE. Wahlweise kann aber auch bei 9i mit einem PFILE initialisiert werden. Jede Instanz hat normalerweise ihr eigenes (S)PFile.

PFILE – initSID.ora

Default Location: `$ORACLE_HOME/dbs`
Das PFILE ist eine ASCII Datei und wird auch als statisches Parameterfile bezeichnet. Man kann es mit einem Texteditor bearbeiten. Die Änderungen werden erst nach dem nächsten Neustart der Instanz wirksam.

PFILE erzeugen: Da ab Oracle 9i kein PFILE mehr per default verwendet wird, muss man es erst erzeugen. Dies ist sinnvoll, da man das PFILE für Sicherungs- und Dokumentationszwecke verwenden kann.

Anmerkung: Die Initialisierung einer Oracle-Instanz ab Version 9i sollte immer mit einem SPFILE erfolgen. Die ausschließliche Verwendung eines PFILES wird nicht empfohlen.

Ausnahmen: Datenbank-Upgrades oder Kompatibilitätsgründe.

Bemerkung: RMAN kann auch das persistente Parameterfile sichern.

```
create PFILE = '/oracle/ora92/database/pfile_testdb29_01.ora'
  from spfile  '/oracle/ora92/database/spifile';
```

SPFILE – spfileSID.ora

Default Location: `$ORACLE_HOME/dbs`
Das SPFile ist binäre Datei und wird auch als persistentes Parameterfile bezeichnet. Änderungen können persistent gegenüber shutdown und startup gemacht werden. Manuell kann es nicht verändert werden. Einsehbar ist es aber z.B. mit dem Unix-Befehl `less spfileSID.ora`.

SPFILE aus einem PFILE erzeugen: Dies kann z.B. bei einem fehlerhaften SPFILE notwendig sein.

```
create SPFILE = '/oracle/ora92/database/spfile_testdb29_01.ora'
  from pfile    '/oracle/ora92/database/inittestdb29.ora';
```

Reihenfolge der Auswertung von Parameterfiles beim Start

1. spfileSID.ora
2. Default SPFILE
3. initSID.ora
4. Default PFILE

Starten mit einem PFILE: Man kann aber auch expizit ein PFILE angeben:

```
startup pfile = '/oracle/ora92/database/pfile_testdb29_01.ora';
```

Parameter ändern: Das SPFILE wird nicht manuel editiert. Die Änderungen erfolgen mittels Anweisungen.

```
ALTER SYSTEM SET parameter_name = parameter_wert
  SCOPE = MEMORY | SPFILE | BOTH SID = '*'
;
```

MEMORY – Änderungen werden nur im Memory gespeichert. Sie sind also nur bis zum Shutdown wirksam.
SPFILE – Änderungen werden nur in das SPFILE geschrieben.
BOTH – Änderungen werden nur im Memory und in das SPFILE geschrieben.
SID – Identifiziert die ORACLE_SID. * bedeutet das Default-SPFILE.

Parameter wieder auf default setzen:

```
ALTER SYSTEM RESET parameter_name parameter_wert
  SCOPE = MEMORY | SPFILE | BOTH
;
```

2.3.3 STARTUP-Optionen

Vergleichbar mit den Runlevel bei vielen Unix-Betriebssystemen ist es auch möglich eine Instanz in Stufen hochzufahren.

1. SHUTDOWN
2. NOMOUNT
3. MOUNT
4. OPEN

Keine Option

```
startup;
```

Wird keine keine Option beim Befehl STARTUP angegeben, starten alle Instanzen entsprechend den Einstellungen in der Datei `/etc/oratab` bis OPEN.

NOMOUNT

```
startup nomount;
```

NOMOUNT wird zur Datenbankerstellung und zur Wiederherstellung der Control Files verwendet. Es werden folgende Schritte abgearbeitet.

- Lesen der Initialisierungsfiles
- Speicherzuweisung für den SGA
- Starten der Backgroundprozesse
- Öffnen der alertSID.log und der Trace Files.

Die Datenbank muss per `DB_NAME` gesetzt sein oder in den Initialisierungsfiles stehen.

MOUNT

```
startup mount;
```

Die Datenbank wird gemountet aber nicht geöffnet. Dies ist für für folgende Arbeiten notwendig.

- Umbennnen/Verschieben von Datafiles
- (De-)Aktivierung von Redo-Log-Archivierungs-Optionen
- Volles Datenbank-Recovery

Es werden folgende Schritte abgearbeitet.

- Assozieren einer Datenbank mit einer vorher gestarteten Instanz,
- Öffnen der Control Files.
- Lesen der Control Files um Namen und Status der Datenfiles und Redo Log Files zu erhalten.
  Die Existens der Datenfiles und Redo Log Files wird aber noch nicht überprüft.

OPEN

```
startup open;
```

Jeder gültige User kann jetzt auf die Datenbank zugreifen. Es werden folgende Schritte abgearbeitet.

- Öffnen der Online Datafiles.
- Öffnen der Redo Log Files.
- Der SMON führt ggf. Instanz Recovery durch.

Weitere STARTUP Optionen

```
startup force;
```

Bricht eine laufende Instanz ab und führt einen normales STARTUP aus.

```
startup restrict;
ORA-01035: ORACLE only available to users with RESTRICTED SESSION privilege
```

Erlaubt nur Usern mit dem Privileg RESTRICTED SESSION den Zugang zur Datenbank.

```
startup recover;
```

Beginnt ein Media Recovery wenn die Datenbank startet.

2.3.4 ALTER DATABASE

```
alter database testdb29 mount;
```

Ändert des Status der Datenbank von NOMOUNT zu MOUNT.

```
alter database testdb29 open read only;
```

Öffnet eine Database schreibgeschützt.

```
alter database testdb29 open read write;
```

Öffnet eine Database mir Schreibzugriff.

2.3.5 ALTER SYSTEM

```
alter system enable restricted session;
```

Erlaubt nur Usern mit dem Privileg RESTRICTED SESSION den Zugang zur Datenbank.

```
alter system disable restricted session;
```

Allen User wird der Zugriff erlaubt.

```
alter system kill session 'SID,SERIAL#';
```

Beendet eine eine Session. SID und SERIAL kann aus V$SESSION ermittelt werden.

```
alter system suspend;
```

```
alter system resume;
```

2.3.6 SHUTDOWN-Optionen

Das Herunterfahren einer Instanz kann mit unterschiedlichen Dringlichkeiten erfolgen.

NORMAL

```
shutdown normal;
```

Neue User können sich nicht mehr anmelden. Bestehende Sessions werden nicht beendet. Erst wenn der letzte User sich abgemeldet hat wird die Instanz herunter gefahren. Database Buffer und Redo Buffer werden in die Files geschrieben. Nach dem Neustart ist kein Recovery notwendig.

TRANSACTIONAL

```
shutdown transactional;
```

Es wird gewartet bis alle offenen Transactionen beendet sind. Ansonsten werden alle Sessions sofort beendet. Nach dem Neustart ist kein Recovery notwendig.

IMMEDIATE

```
shutdown immediate;
```

Laufende Transaktionen werden zurückgerollt. Aktuelle SQL Statements werden nicht beendet. Alle User Sessions werden sofort beendet. Nach dem Neustart ist kein Recovery notwendig.

ABORT

```
shutdown abort;
```

Transaktionen werden weder beendet noch zurückgerollt. Dateien werden nicht geschlossen. Die Datenbank wird weder geschlossen noch dismountet. Die Datenbank ist inkonsistent. Beim Neustart erfolgt ein Rollback aller Transactionen Beim Neustart ist ein Instanz-Recovery notwendig. Dies erledigt SMON.

2.3.7 Instance Recovery

Instance Recovery – Bei Oracle Real Application Clusters Konfiguration.
Crash Recovery – Bei nur einer Instance.

Reihenfolge beim Instance Recovery

1. Nicht synchronisierte Datendatei (Crash).
2. Roll Forward (Redo)
   DBWR schreibt festgeschriebene und nicht festgeschriebene Daten in die Datendateien. Alle Änderungen, die in den Log-Dateien protokolliert wurden, werden auf die Datenblöcke angewendet.

3. Festgeschriebene und nicht festgeschriebene Daten befinden sich in den Datendateien. Die Datenbank wird geöffnet.
4. Rollback (Undo).
5. In Dateien festgeschriebene Daten.

2.3.8 Überwachen der Alert Log File und Trace Files

Zur Fehlerüberwachung und Diagnostik dienen mehrere Arten von Log/Trace-Files. Diese ASCII-Dateien können mit den üblichen Unix-Befehlen (`tail`, `grep`) betrachtet und ausgewertet werden.

alertSID.log

Pfad wird in BACKGROUND_DUMP_DEST festgelegt. Dies ist die eigentliche Log-Datei für die Instanz. Diese sollte regelmäßig untersucht und dann geleert werden.

Background Trace Files

Pfad wird in BACKGROUND_DUMP_DEST festgelegt. Die Bennenung ist: `sid_prozessname_pid.trc`. Wobei der Prozessname sich aus dem Background-Prozess ergibt (LGWR, DBWR, ...). Die PID ist die PID vom Betriebssystem. Diese Dateien werden nur bei Fehlern angelegt. Diese sollte regelmäßig untersucht und dann gelöscht werden.

User Trace Files

Pfad wird in USER_DUMP_DEST festgelegt. Die Bennenung ist: `sid_ora_PID.trc`. Diese Dateien werden nur bei Fehlern angelegt. Diese sollte regelmäßig untersucht und dann gelöscht werden.

(De-)Aktivieren des User-Tracing

```
alter session set SQL_TRACE = TRUE;
```

Achtung: Es entstehen sehr viele Daten, da z.B. jedes SQL-Statement geloggt wird. Die Default-Einstellung ist TRUE und sollte daher auf FALSE gesetzt werden.

2.3.9 Übung PFILE / SPFILE

Übungen siehe Seite 261.

2.3.10 Übung Hoch- / Herunterfahren

Übungen siehe Seite 261.

2.4 Erstellen einer Datenbank

Unix-Umgebungsvariablen

```
export ORACLE_HOME=/opt/oracle/OraHome1
export ORACLE_SID=TESTDB29
export PATH=$PATH:$ORACLE_HOME/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
```

2.4.1 Optimal Flexible Architecture (OFA)

Oracle empfielt das Standard Datenbank Architektur Layout. Regeln:

- Die Verzeichnisstruktur ermöglicht jedes Database Files in jeder Festplattenresource zu speichern.
- Unterschiedliche Objekte mit unterschiedlichen Eigenschaften werden in unterschiedliche Tablespaces gespeichert.
- Maximale Datenbank Zuverlässigkeit und Performance durch Aufteilung der Datenbankkomponenten auf unterschiedliche Festplattenresourcen.

2.4.2 CREATE DATABASE

```
CREATE DATABASE [database]
{ CONTROLFILE REUSE
| LOGFILE [GROUP integer] filespec [, [GROUP integer] filespec]...
| MAXLOGFILES integer
| MAXLOGMEMBERS integer
| MAXLOGHISTORY integer
| MAXDATAFILES integer
| MAXINSTANCES integer
| { ARCHIVELOG | NOARCHIVELOG }
| CHARACTER SET charset
| NATIONAL CHARACTER SET charset
| DATAFILE filespec [autoextend_clause] [, filespec [autoextend_clause]]...
| default_temp_tablespace
| undo_tablespace_clause
| SET STANDBY DATABASE { PROTECTED | UNPROTECTED }
| set_time_zone_clause
}...
```

2.4.3 Datenbankarten

Typical

- Online Transaction Processing (OLTP)
- Data Warehousing
- Mixed

Custom

Einstellungen werden von erfahrenen Datenbankadministratoren individuell vorgenommen.

2.4.4 Oracle Managed Files (OMF)

Mittels OMF wird die File-Administration auf OS-Ebene vereinfacht, da Oracle die Dateinamen nach festen Regeln vergibt.

Control Files – ora_*.ctl
Redo Log Files – ora_*.log
Data Files – ora_*.dbf
Temporary Data Files – ora_*.tmp

OMF wird durch Definition der folgenden Initialisierungsparameter aktiviert. Beispiel:

```
DB_CREATE_FILE_DEST     = /$HOME/ORADATA/u05
DB_CREATE_ONLINE_DEST_1 = /$HOME/ORADATA/u01
DB_CREATE_ONLINE_DEST_2 = /$HOME/ORADATA/u02
 ...
```

2.4.5 Problembehebung

Mitunter kann es vorkommen, dass es durch falsche Bedienung zu Fehlern kommt.

```
ORA-00604: rekusiver SQL-Fehler \\
ORA-06553-213: Auf Standard-Paket besteht kein Zugriff
```

Hier hilft das Ausführen des folgenden Scripts.

```
ORACLE_HOME/rdbms/admin/standard.sql
```

2.5 Data Dictionary, Dynamic Performance Views

Data Dictionary

Die Data Dictionary Views werden mit der Datenbank erstellt und enthalten Beschreibungen der Objekte der Datenbank.

Data Dictionary View Categories

DBA_* : Was ist in allen Schemen?

```
select owner, object_name, object_type from dba_objects;
```

Gibt viele Informationen alle Objekte der Datenbank zurück.

```
select * from dictionary where table_name like 'dba_seg%';
describe dba_users;
select * from dba_users;
```

ALL_* : Worauf kann der User zugreifen?

```
select owner, object_name, object_type from all_objects;
```

Gibt viele Informationen aller Objekte aus, auf die der User zugreifen kann.

USER_* : Was ist im Schema des Users?

```
select object_name, object_type from user_objects;
```

Gibt viele Informationen alle Objekte aus, die zu den Schema des Users gehören.

Data Dictionary Examples

Übersicht über die vielen Data Dictionary Views.

```
select * from dictionary where table_name like 'v$data%';


DICTIONARY
names of all the data dictionary views

DBA_OBJECTS
number of objects a user has created

DBA_TABLES >> NEXT_EXTENT, PCT_INCREASE
size of 3rd extent (dictionary managed tablespace)

DBA_CONSTRAINTS
foreign key constraint and the referenced primary key

DBA_INDEXES
```

```
Index names, Index types

DBA_FREE_SPACE
free space in a certain tablespace

DBA_SEGMENTS
space allocated for temporary segments /
location of all the tables and indexes owned by one user

DBA_TAB_PRIVS
object privileges for all database users

DBA_USERS
all database users, default settings, account expired,
default tablespace name, profile name
```

2.5.1 Dynamic Performance Views

Die Dynamic Performance Views enthalten aktuelle Informationen für den Datenbankadministrator (DBA) zur Überwachung und Tuning der Datenbank und der Instanz. Die Dynamic Performance Views sind übersichtlichere Darstellungen aus den Base Tables, in denen die Informationen in kryprischer Form gespeichert werden.

Dynamic Performance Views Examples

Von `v$fixed_table` erhält man auch ein Listing der Dynamic Performance Views.

```
select * from v$fixed_table;


describe v$instance;
```

Anzeige des Inhaltes der View.

```
select * from v$instance;


select name from v$parameter;


SELECT name, status FROM v$controlfile;
number, names, status, and location of the control files

V$LOG, V$THREAD, V$DATAFILE
obtain information from the control file

V$CONTROLFILE_RECORD_SECTION
data file maximum when database was created

V$SORT_USAGE
disk sorts on database right now
```

2.6 Control Files

Control Files sind kleine binäre Dateien, die den aktuellen Status der Datenbank definieren. Sie sind für die Intentigrität der Datenbank verantwortlich. Ein Verlust der Control Files erfordern ein Recovery der Datenbank. Beim MOUNT-Status werden die Control Files gelesen.

Der Inhalt der Control Files besteht aus

- Datenbankname und Identifier
- Time Stamp der Datenbank-Erzeugung
- Tablespace Namen
- Namen und Pfade der Data Files und Redo Log Files
- Aktive Redo Log Sequenz Nummer
- Checkpoint Informationen
- Anfang und Ende des Undo-Segments
- Redo Log Archive Information
- Backup Information

2.6.1 Vervielfachen der Control Files

Da Control Files sehr wichtig sind, sollte man die Control Files spiegeln. Man kann bis zu 8 identische Control Files auf unterschiedlichen Festplatten verwalten. Bei einem Festplattenkrasch kann man ein anderes Control File bei heruntergefahrener Instanz an den ursprünglichen Ort der zerstörten Datei kopieren und mit dieser Datei neu starten.

Control Files müssen beim Erstellen der Datenbank mit angegeben werden. Ansonsten kann keine Datenbank erstellt werden. Die Lage und Namen de Control Files wird in den Dateien SPFILE und PFILE gespeichert.

2.6.2 Ändern des SPFILE für verschobene CONTROL-Files

```
alter system
  set control_files = '/oracle/oradata/testdb29/CONTROL01.CTL',
  '/hdd2/control02.ctl',
  '/hdd3/control03.ctl'
  SCOPE = SPFILE
;
shutdown immediate;
startup;
```

2.6.3 Informationen zu den Control Files

```
show parameter control_files;
select name,status from V$CONTROLFILE;
select name,value from V$PARAMETER where name = 'control_files';
```

2.6.4 Übung Control Files spiegeln

Übungen siehe Seite 262.

2.7 Redo Log Files

Redo Log Files speichern alle Änderungen an den Daten. Sie sind für den Recovery-Mechanismus zuständig. Redo Log Files können in Gruppen organisiert werden. Da die Redo Log Files abwechselnd geschrieben werden, sind mindestens zwei Gruppen notwendig.

2.7.1 Online-Redo-Logs

Anzeige der Gruppe, Namen, Status

```
select group#,member,status from V$logfile;
```

Anzeige Gruppe und Status

```
select group#,status from V$log;
```

Member hinzufügen

```
alter database
  add logfile member
    '/oracle/oradata/testdb29/redo01a.LOG'
    to group 1
;
```

Neue Gruppe hinzufügen

```
alter database
  add logfile group 4
   ('/oracle/oradata/testdb29/redo04a.LOG',
    '/oracle/oradata/testdb29/redo04b.LOG')
  size 100M
;
```

Log-Files umschalten

```
alter system switch logfile;
```

Checkpoint forcieren

```
alter system checkpoint;
```

Gruppe löschen / Größe ändern

Da das Modifizieren (z.B. Größe ändern) von Gruppen nicht möglich ist, müssen diese gelöscht werden. Gruppe darf dabei nicht CURRENT oder AKTIVE sein. Files werde nicht physisch gelöscht.

```
alter database drop logfile group 4;
```

Member löschen

Der letzte Member einer Gruppe darf nicht gelöscht werden. Die Gruppe darf beim Löschen nicht CURRENT oder AKTIVE sein. Files werden nicht physisch gelöscht.

```
alter database drop logfile member '/oracle/oradata/testdb29/redo01a.LOG';
```

2.7.2 Offline-Redo-Logs (Archive Redo Logs)

Der Archivierungsmodus ist normalerweise deaktiviert.

```
select log_mode from V$database;
LOG_MODE
------------
NOARCHIVELOG


select archiver from v$instance;
```

Gespeichert wird per default in `/oracle/ora92/rdbms/ARCn*` Für diesen Vorgang ist MOUNT EXCLUSIVE und die geschlossene DB erforderlich.

Dienst starten:

```
alter system set LOG_ARCHIVE_START=true SCOPE=SPFILE;
shutdown immediate;
startup mount;
```

Archivierung aktivieren:

```
alter database archivelog;
alter database open;
```

2.7.3 Übung Redo Log Dateien

Übungen siehe Seite 264.

2.8 Tablespaces, Datafiles

Oracle speichert Daten in Tablespace (logisch) und in Data Files (physisch). Ein Tablespace kann nur zu einer Datenbank gehören. Ein Tablespace kann ein oder mehrere Data Files beinhalten. Ein Data File kann nur zu einem Tablespace und zu einer Datenbank gehören.

2.8.1 Arten von Tablespaces

System Tablespace

- Wird mit der Datenbank erstellt.
- Enthält das Data Dictionary.
- Enthält das System Undo Segment.

Non-System Tablespace

- Separate Segmente.
- Einfache Speicherplatz-Administration.
- Steuerung der Speichergrößen für die User.

2.8.2 Anzeige der Tablespaces

```
select * from dba_tablespaces;
```

2.8.3 Tablespace erzeugen

```
create tablespace testus
  datafile '/var/tablespaces/testus.dbf'
  size 100M autoextend on next 5M maxsize 500M
;
```

2.8.4 Locally Managed Tablespace

Der Locally Managed Tablespace ist erst ab 9i Default-Einstellung.

```
create tablespace testus_l
  datafile '/var/tablespaces/testus_l.dbf'
  size 100M extent management local
;
```

2.8.5 Dictionary Managed Tablespace

Der Dictionary Managed Tablespace wird inzwischen sehr selten verwendet.

```
create tablespace testus_d
  datafile '/var/tablespaces/testus_d.dbf' size 100M
  extent management dictionary
  default storage
   (initial 32k next 64k minextents 5 maxextents unlimited pctincrease 50)
;
```

Wenn der SYSTEM-Tablespace als Locally Managed Tablespaces angelegt wurde, kann weiterer Tablespace nur locally angelegt werden.

2.8.6 Undo Tablespace (Before-Images)

Der Undo Tablespace gilt für gesamte DB und kann keinem User zugewiesen werden.

```
create undo tablespace undo_01
  datafile '/var/tablespaces/undo01.dbf' size 50M;
```

2.8.7 Temporary Tablespace

Der Temporary Tablespace wird für SORT gebraucht. Nur ein ein temporary Tablespace kann pro DB aktiv sein (default). Dieser sollte wachsen können (d.h. auf extra Platte legen), er schrumpft aber nicht mehr.

```
create temporary tablespace temp01
  tempfile '/var/tablespaces/temp_01.dbf' size 500M
  extent management local uniform size 4M
;
```

Anzeige des Default Temporary Tablespace:

```
select * from database_properties
  where PROPERTY_NAME = 'DEFAULT_TEMP_TABLESPACE';
```

Wechsel des Default Temporary Tablespace:

```
alter database default temporary tablespace temp01;
```

Beachte: Einen temporären Tablespace kann man keine Quota für User zuweisen.

2.8.8 Tablespace offline setzten

Notwendig für Recovery

```
alter tablespace testus_l offline normal;
```

- normal – Alle Data Blocks werden von aus den SGA in die Tabelespaces geschrieben (default).
- temporary – Bewirkt einen Checkpoint. Alle Offline Files erfordern ggf. Media Recovery.
- immediate – Stellt nicht sicher das alle Tablespace Files verfügbar sind. Es wird kein Checkpoint erzwungen. Media Recovery ist unbedingt notwendig. Daher sollt immediate vermieden werden.
- for recover – Bringt Tablespace offline für Tablespace Point-In-Time-Recovery.

2.8.9 Tablespace online setzten

```
alter tablespace testus_l online;
```

2.8.10 Tablespace Read Only setzen

Der Tablespace muss dazu online sein.

```
alter tablespace testus_l read only;
```

2.8.11 Welche Tablespaces sind autoextensible?

Anzeige der Tablespaces, die sich automatisch vergrößern können.

```
select tablespace_name, file_name, autoextensible
  from DBA_DATA_FILES;
```

2.8.12 Datendatei vergrößern / Datendatei hinzufügen

Automatische Erweiterungen bewirken Performanceverluste.

Datendatei vergrößern:

```
alter database datafile '/var/tablespaces/testus.dbf'
  resize 200M;
```

2.8.13 Eine Datendatei hinzufügen

```
alter tablespace testus
  add datafile '/var/tablespaces/testus_longus.dbf'
  size 200M autoextend on next 100M;
```

2.8.14 Datafiles zu einem anderen Tablespace verschieben

Tablespace offline setzen.

```
alter tablespace testus_l offline normal;
```

Datei mit Betriebssystembefehlen verschieben.

```
alter tablespace testus_l
  rename datafile '/var/tablespaces/testus_l.dbf'
    to '/hdd2/testus_l.dbf'
;
alter tablespace testus_l online;
```

Tablespace online setzen.

```
alter tablespace testus_l online;
```

2.8.15 Löschen von Tablespace

Nur wenn der Inhalt nicht mehr gebraucht wird. Beachte Constraints und andere Beziehungen auf diesen Tablespace.

```
drop tablespace testus_l including contents
  and datafiles cascade constraints;
```

2.9 Storage Strukturen und Beziehnungen

2.9.1 Beispiel: Wie gross ist das 5. Element?

```
CREATE  TABLE  hr.departments
  (
    department_id NUMBER(4),
    department_name VARCNAR2(30),
    manager_id NUMBER(6),
    location_id NUMBER(4)
  )
  STORAGE
  (
    INITIAL     200K
    NEXT        200K
    PCTINCREASE  50
    MINEXTENTS    1
    MAXEXTENTS    5
  )
  TABLESPACE  data
;
```

1. 200K (INITIAL)
2. 200K (NEXT)
3. 300K (Letztes * PCTINCREASE %)
4. 450K (Letztes * PCTINCREASE %)
5. **675 K**

2.9.2 Data Block Management

Automatic Segment Space Management

Der Automatic Segment-Space Management ist eine Methode zum verwalten des freien Platzes innerhalb von Database Segments. Dabei werden Bitmaps und keine Free Lists verwendet.

- Einfache Verwendung
  PCTUSED, FREELIST, FREELIST GROUPS werden automatisch verwaltet.
- Bessere Platzausnutzung
- Bessere Handling bei Merhfachzugriffen

```
CREATE TABLESPACE data02
  DATAFILE '/u01/oradata/data02.dbf' SIZE 5M
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K
  SEGMENT SPACE MANAGEMENT AUTO
;
```

Bei 'SEGMENT SPACE MANAGEMENT AUTO' werden Angaben zu PCTUSED, FREELIST, FREELIST GROUPS ignoriert.

Hinweis: Bei einem Tablespace mit Automatic Segment Management wird der der freie Speicher im Undo Tablespace verwaltet.

Manual Segment Space Management

Das Manual Segment Space Management ermöglicht die Konfiguration von PCTFREE, PCTUSED, FREELIST. Es war die einzige Methode bis Oracle 8i. Das Manual Segment Space Management ist die Default-Methode in Oracle 9i.

2.10 Undo Management

Das Undo Management ist für ROLLBACK und Lesekonsistenz notwendig. Das Before-Image wird nach COMMIT verworfen.

2.10.1 Undo Tablespace

Der Undo-Tablespace wird automatisch erstellt. Dieser Bereich wurde in Oracle 9i komplett neu programmiert. Das Undo wird nun automatisch abgesichert. Dafür ist PMON verantwortlich. Man kann mehrere Undo-Tablespaces erstellen. Einer kann aber nur aktiv sein. Undo-Segmente hiessen früher Rollback-Segemente.

Hinweis: Bei einem Tablespace mit Automatic Segment Management wird der der freie Speicher im Undo Tablespace verwaltet.

Erstellung

```
create undo tablespace undotbs2
  datafile '/var/tablespaces/undotbs2.dbf'
  size 500M autoextend on
;
```

Modifizieren

Eine Datei hinzufügen.

```
alter tablespace undotbs2
  add datafile '/var/tablespaces/undotbs2a.dbf'
  size 500M autoextend on
;
```

Dynamischer Wechsel des aktiven Undo-Tablespace (z.B. bei Plattenwechsel). Dabei werden alte Transaktionen nicht auf den neuen Undo-Tablespace verschoben.

```
alter system set undo_tablespace = undotbs2;
```

Anzeige Status

```
select segment_name,tablespace_name,status from dba_rollback_segs;
```

Zuerst ist der STATUS = PENDING, da alte Transaktionen noch abgeschlossen werden müssen.

Undo-Tablespace löschen

Zuvor ist neuer Undo-Taplespace zu erstellen und zu aktivieren.

```
drop tablespace undotbs2;
```

Anzeige ob offene Transaktionen existieren.

```
select a.name,b.status
  from v$rollname a, v$rollstat b
  where a.name IN
   (
     select segment_name
        from dba_segments
        where tablespace_name = 'UNDOTBS1'
   )
    and a.usn = b.usn
;
```

2.11 Tabellen

2.11.1 Storing User Data

- Regular Table
  Dies ist die am meisten verwendete Tabellenart. Ein Datenbankadministrator hat nur wenig Einfluss auf die Speicherung der Datensätze.
- Partitioned Table
  Diese Art ermöglicht die Bildung von skalierbaren Applikationen. Jede Partition kann in unterschiedlichen Tablespaces gespeichert werden.
- Index-Organized Table
  Schneller Key-basierter Zugriff.
- Clustered Table
  Wird aus einer oder mehreren Tabellen gebildet, die sich den gleichen Datenblock teilen.

Von den vier Tabellentypen wird hier die Regular Table eingehender besprochen.

2.11.2 ROWID

```
select dummy,rowid from dual;
```

2.11.3 Tabelle in einem Tablespace erstellen

```
create table bla ( s1 varchar2(29) ) tablespace testus;
```

2.11.4 Tabelle zu einem anderen Tablespace verschieben

Das Verschieben einer Tabelle von einem Tablespace in einem anderen sollte nur in Ausnahmefällen erfolgen. Dies kann z.B. notwendig sein, wenn die Performance drastisch sinkt. Sämtliche Indizes dieser Tabelle werden gelöscht und müssen neu erstellt werden.

```
alter table bla move tablespace testus;
```

2.11.5 Anzeige des Tablespace einer Tabelle

```
describe user_tables;
select table_name, tablespace_name
  from user_tables;
```

2.11.6 Temporäre Tabelle erstellen

Temporäre Tabellen können z.B. für Zwischenberechnungen verwendet werden. DML Locks werden dafür nicht verwendet.

```
create global temporary table mein_temp
  on commit delete rows as
  select * from dual
;
insert into mein_temp values ('3');
select * from mein_temp;
-- Es wird ein Dateinsatz angezeigt.
commit;
select * from mein_temp;
-- Es wurden keine Zeilen ausgewählt
```

2.11.7 Spalte UNUSED setzen

Eine Tabellenspalte auf UNUSED setzen geht schneller als DROP Spalte. Dies ist aber als SYS nicht möglich.

```
create table blabla ( s1 varchar2(29), s2 varchar2(29) );
alter table blabla
  set unused column s2 cascade constraints;
describe blabla;
alter table blabla
  drop unused columns checkpoint 1000;
alter table blabla
  drop columns continue checkpoint 1000;
```

2.11.8 Anzeige der UNUSED Spalten

Einloggen als User sys.

```
select * from dba_unused_col_tabs;
select * from dba_partial_drop_tabs;
```

2.11.9 Anzeige von Informationen zu Tabellen

```
describe dba_tables;
select * from dba_tables
  where table_name = 'DUAL';
select table_name, tablespace_name from dba_tables
  where owner = 'SYS';
select object_name, created from dba_objects
  where object_name like 'EMPLOYEES';
```

2.12 Indizes

Ein Index ist eine Baumstruktur, die einen direkten Zugriff auf ein Datensatz einer Tabelle ermöglicht. Indizes solten immer immer in einem separaten Tablespace gespeichert werden, der auf einer separaten Platte liegen sollte.

2.12.1 Arten

Logisch

- Über eine Spalte oder über mehrere Spalten (Concatenated, max. 32)
- Unique oder Nonuique
- Auf Funktionen basiert (B-Tree oder Bitmap)
- Domain (Applikationsspezifisch) für Scalar-, Objekt- oder LOB-Datentypen, max.1 Spalte

Physikalisch

- Partitioniert oder Nonpartitioniert
  Verwendung für große Tabellen,
  speicherbar über mehrere Tablespaces,
  meist verwendet mit partitionierten Tabellen
- B-Tree
  - Normal oder Reverse Key
- Bitmap

Eingehender wird hier auf den B-Tree- und Bitmap-Index eingegangen.

B-Tree

- Günstig bei vielen unterschiedliche Werten (hohe Kardinalität), besonders bei UNIQUE.
- Sehr uneffizient bei SELECT-Abfragen mit OR.
- Günstig für OLTP.
- NOT NULL kann nicht indiziert werden.

Normal: Der normale B-Tree wird am meisten verwendet und wird nachfolgend eingehender behandelt.

Reverse Key Index: Beim Erzeugen eines Reverse Key Index wird der Wert in der Key-Spalte 'umgedreht'.

```
12345 --> 54321
12346 --> 64321
12347 --> 74321
```

Der Reverse Key Index wird sehr selten verwendet. Meist um bei Sequenzen mit kontinuierlich folgenden Werten ein Verteilen auf unterschiedliche Blöcke zu erreichen.

Beachte: Wenn Applikationen Wertebereiche festlegen, kann der Reverse Key Index nicht verwendet werden.

Bitmap

- Bei geringer Kardinalität (<10.000), also bei wenig unterschiedlichen Werten (z.B. männlich, weiblich).
- Ein Update einer Key-Spalte ist sehr aufwendig.
- Effizient bei SELECT-Abfragen mit OR.
- Günstig für Warehousing.
- Sinnvoll bei DSS
- NOT NULL kann indiziert werden.

2.12.2 Index erstellen

Empfehlungen

- Balance zwischen Abfragegeschwindigkeit und DML Bedürfnissen.
  Während die Abfragegeschwindigkeit sich erhöht, sinkt die Geschwindigkeit bei DML-Befeheln.
- Indizes sollten in separaten Tablespace und Festplatten gespeichert werden.
- Höhrere Geschwindigkeiten wird bei großen Inizes durch NOLOGGING (kein Redo) erreicht.
- Meist haben Indizes mehr Einträge pro Block als eine Tabelle.
  INITRANS sollte daher höher als die zugehörige Tabelle sein.

B-Tree

Normal

```
create index id_dual on dual (dummy)
  tablespace testus;
```

Der B-Tree ist der Default Index und braucht daher nicht angegeben zu werden.

```
CREATE INDEX scott.emp_lname_idx
  ON scott.employees(last_name)
  PCTFREE 30
  STORAGE(INITIAL 200K NEXT 200K
  PCTINCREASE 0
  MAXEXTENTS 50)
  TABLESPACE indx01
;
```

Reverse Key Index

```
CREATE UNIQUE INDEX scott.ord_ord_no_idx
  ON scott.ord(ord_no) REVERSE
  PCTFREE 30
  STORAGE(INITIAL 200K NEXT 200K
  PCTINCREASE 0
  MAXEXTENTS 50)
  TABLESPACE indx01
;
```

Erstellen eines Funktionsindex: Ein Funktionsindex beinhaltet eine Funktionn. Oft ist dies `lower()`, um Groß- und Kleinschreibung zu ignorieren.

Achtung: Query Rewrite muss vorher gegeben worden sein.

```
CREATE INDEX scott.funct_ind1
  ON scott.emp(lower(ename))
  PCTFREE 30
  STORAGE(INITIAL 200K NEXT 200K
  PCTINCREASE 0
  MAXEXTENTS 50)
  TABLESPACE indx01
;
```

Bitmap

Beispiel mit Storage Parametern.

```
create bitmap index id_dual
  on dual (dummy)
  pctfree 30 storage(initial 200k next 200k pctincrease 0 maxextents 50)
  tablespace testus
;
```

Diese Werte (initial, next, pctincrease, maxextents) werden in diesem Beispiel ignoriert, da der Tablespace testus als Local Managed definiert wurde.

2.12.3 Ändern der Storage Parameter eines Index

Die zu ändernden Parameter sind die gleichen wie bei Tabellen. Meist wird MAXEXTENTS erhöht.

```
alter index id_dual storage(next 400k maxextents 100);
FEHLER in Zeile 1:
ORA-25150: ÄNDERN von Extent-Parametern nicht zulässig
```

Dies geht nicht, da der Tablespace hier Local Managed ist.

2.12.4 Index verschieben

Dies ist nicht sehr zu empfehlen, da es sehr lange dauert.

```
alter index id_dual rebuild tablespace testus;
```

2.12.5 Index reorganisieren

REBUILD – Reorganisieren Online

REBUILD ONLINE dient zur staleness Korrektur von Index. REBUILD ONLINE ist sehr zeitaufwendig. Umfangreiche DML-Operationen sind dabei nicht zu empfehlen. DDL-Operationen werden für diese Zeit gesperrt. Restriktionen:

- Indizes von temporäre Tabellen können nicht mit REBUILD reorganisiert werden.
- Partitionierte Indizes müssen einzeln mit REBUILD reorganisiert werden.
- Man kann nicht ungenutzten Speicher freigeben.
- Man kan nicht den Wert von PCTFREE für den gesamten Index ändern.

```
alter index id_dual rebuild online;
FEHLER in Zeile 1:
ORA-08108:
Dieser Index-Typ kann nicht online erstellt oder neu erstellt werden.
```

Zu klein?

COALESCE – Index-Zusammenführung

Um den Index neu zu organisieren kann bei B-Tree-Index COALESCE angewendet werden. Dabei können Abfragen von Usern, die den Index verwendet, ausgeführt werden. Wird laut Dozent kaum verwendet (ausser in der Prüfung). Besser ist hier Rebuild Online.

```
alter index id_dual coalesce;
```

2.12.6 Index logisch überprüfen

ANALYZE erzeugt die View `index_stats`, die dann abgefragt werden kann.

```
analyze index id_dual validate structure;
select name, blocks, pct_used, distinct_keys lf_rows, del_lf_rows
  from index_stats
;
NAME         BLOCKS  PCT_USED  LF_ROWS  DEL_LF_ROWS
-------------------------------------------------
ID_DUAL          32         1        1            0
```

Wenn viele gelöschte Leafs (DEL_LF_ROWS >10%) angezeigt werden, sollte neu organisiert werden.

2.12.7 Statistiken über die Nutzung des Index

Ab Oracle9i können Statistiken über die Nutzung eines Index erzeugt werden. Diese werden in V$OBJECT_USAGE angezeigt. Nicht genutzte Indizes können so erkannt und gelöscht werden.

Monitoring starten

```
alter index hr.dept_id_idx monitoring usage;
```

Monitoring stoppen

```
alter index hr.dept_id_idx nomonitoring usage;
```

2.12.8 Informationen über Indizes

- DBA_INDEXES – Indexname und -typ
- DBA_IND_COLOUMNS – indizierte Spalten
- V$OJECT_USAGE – Statistiken über Indexnutzung

2.12.9 Index löschen

Bevor man große Datenmengen importieren will, sollte man den Index löschen und nach dem Import neu erstellen. Dies geht schneller. Automatisch erstellte Indizes können erst nach dem Deaktivieren der entsprechenden Constraints gelöscht werden.

Löschen Sie Indizes, die selten gebraucht werden und erstellen Sie diese kurz vor Gebrauch.

```
drop index id_dual;
```

2.12.10 Views

USER_INDEXES

Infos über Indizes

USER_IND_COLUMNS

Infos über die Spalten

2.12.11 Übung Index

Übungen siehe Seite 265.

2.13 Datenintengrität

2.13.1 Constraint Stati

- disable novalidate
  Das Constraint wird nicht getestet. Neue und vorhandene Werte können gegen die Regeln des Constraint verstossen.
- disable validate
  Jegliche Änderungen in der Constraint Spalte sind nicht erlaubt.
- enable novalidate
  Neue Daten müssen den Constraint-Regeln entsprechen. Vorhandene Daten werden aber nicht überprüft.
- enable validate
  Sowohl neue Daten als auch vorhandene Daten müssen den Constraint-Regeln entsprechen.

DISABLE

Der Primary Key sollte nicht ausgeschaltet werden, da der entsprechende Index gelöscht wird.

```
alter table bla DISABLE | NOVALIDATE CONSTRAINT ck_bla;
```

ENABLE

```
alter table bla ENABLE | NOVALIDATE CONSTRAINT ck_bla;
```

Dies kann sehr lange dauern, da die gesamte Tabelle überprüft wird. Gibt es eine Regelverletzung in der Tabelle kommt es zu einer Fehlermeldung und der Befehl kann nicht ausgeführt werden. Leider werden die Regelverstöße nicht angezeigt.

2.13.2 Exceptions-Tabelle

Die Exceptions-Tabelle zeigt Regelverstöße an. Erzeugt wird diese mit `/oracle/ora92/rdbms/admin/utlexpt1.sql`

```
-- Inhalt von utlexpt1.sql:
create table exceptions
  (
   row_id urowid,owner varchar2(30),
   table_name varchar2(30),
   constraint varchar2(30)
  )
;

alter table bla
  enable validate constraint ch_bla
  exceptions into exeception
;
```

Nun werden alle Regelverstöße in die Tabelle exeception gespeichert.

2.13.3 Verzögerte (deferred) CONSTRAINTs

Die Regelüberprüfung erfolgt erst beim Ende einer Transaktion, z.B. bei COMMIT. Angabe bei CONSTRAINT-Erstellung.

```
create table test1 (name varchar2(20)
  constraint ch_test1 check (name not in 'Anna')
  deferrable initially deferred);
```

Hier ist die Verzögerung möglich aber am Anfang deaktiviert.

2.13.4 Übung Constraints

Übungen siehe Seite 267.

2.14 Password Security and Resources

2.14.1 Profile

Per Profile erfolgt die Passwortverwaltung (Länge, Alterung, ...). Das Default-Profile erhält jeder User nach dessen Erstellung. Das Default-Profile ist nicht löschbar.

Profile erstellen

```
create profile p_nutzer
  limit failed_login_attempts 3
  password_lock_time 2/24
  password_life_time 30
  password_grace_time 5
  password_reuse_time 366
  password_reuse_max 1
;
```

Beispiel: Ein User versucht sich fünf mal mit falschen Passwort einzuloggen. Wie lange muss er warten, bis er einen neuen Versuch starten kann?

```
ALTER PROFILE DEFAULT LIMIT
  PASSWORD_LIFE_TIME 60
  PASSWORD_GRACE_TIME 10
  PASSWORD REUSE TIME 1800
  PASSWORD_REUSE_MAX UNLIMITED
  FAILED_LOGIN_ATTEMPTS 5
  PASSWORD_LOCK_TIME 1/1440
  PASSWORD_VERIFY_FUNCTION verify_function
;
```

Er muss eine Minute warten. PASSWORD_LOCK_TIME wird in Tagen angegeben. Ein Tag hat 1440 Minuten (60*24).

Profile zuweisen

Profile können nicht anderen Profilen und Rollen zugewiesen werden. Eine Zuweisung hat keine Auswirkunge auf aktuelle Sessions.

```
alter user hans profile meister;
```

Ein Profile ändern

```
alter profile p_nutzer limit failed_login_attempts 4;
```

VERIFY_FUNCTION

```
/oracle/ora92/rdbms/admin/utlpwdmg.sql
```

Profile löschen

```
drop profile p_nutzer;
```

2.14.2 Resource Management

Das Resource Management dient zur Begrenzung von Ressourcen. Diese können auf Session Level, auf Call Level oder beiden zugewiesen werden.

Aktivieren des Resource Managements

```
alter system set resource_limit = true;
```

Session Level

- CPU_PER_SESSION – Totale CPU-Zeit in hundertstel Sekunden.
- SESSIONS_PER_USER – Maximale Anzahl von konkurierenden Session für einen User.
- CONNECT_TIME – Verstrichende Connection-Zeit in Minuten.
- IDLE_TIME – Inaktive Zeit in Minuten.
- LOGICAL_READS_PER_SESSION – Maximale Anzahl der gelesenen Datenblöcke.
- PRIVATE_SGA – Privater Speicherplatz im SGA in bytes (nur Shared Server).

```
create profile p_nutzer
  limit connect_time 1430
  idle_time 30
  sessions_per_user 3
;
```

1/1440 = 1 Minute

Call Level

- CPU_PER_CALL – CPU-Zeit pro Aufruf in hundertstel Sekunden.
- LOGICAL_READS_PER_CALL – Maximale Anzahl der gelesenen Datenblöcke.

Anzeige User-Stati

(OPEN, EXPIRED LOCKED, ...)

```
select username, password, account_status from dba_users
;
select * from dba_profiles
  where resource_type = 'Password'
    and profile = 'default'
;
```

2.15 User

2.15.1 Schema

Ein Schema ist eine benannte Sammlung von Objekten. Wenn ein User erstellt wird, wird auch das gleichnamige Schema erstellt. Das Schema erscheint erst wenn Objekte in dem Schema angelegt werden. Die Begriffe User und Schema werden oft gleichwertig benutzt.

2.15.2 User und Schema erstellen

Welchen Tablespace soll der User haben?

- Nicht den System-Tablespace! Immer einen Default-Tablespace angeben!
- Welchen temporären Tablespace soll der User haben?
- Welche Quota für welche Tablespaces?
- Welche Profile?
- Welche Privilegien?
- Welche Rollen?

```
CREATE USER user
  IDENTIFIED {BY passwort | EXTERNALLY}
  [ DEFAULT TABLESPACE tablespace ]
  [ TEMPORARY TABLESPACE tablespace ]
  [ QUOTA {integer [K | M] | UNLIMITED } ON tablespace
  [ QUOTA {integer [K | M] | UNLIMITED } ON tablespace
  ] ... ]
  [ PASSWORT EXPIRE ]
  [ ACCOUNT { LOCK | UNLOCK} ]
  [ PROFILE { profile | DEFAULT } ]
;
```

BY passwort – Die Authentifizierung erfolgt mittels Passwort.
EXTERNALLY – Die Authentifizierung durch das Betriebssystem.
GLOBALLY AS – Die Authentifizierung erfolgt global (Enterprise Directory Service).
DEFAULT—TEMPORARY TABLESPACE – Definiert den Default bzw. Temporary Tablespace.
QUOTA – Legt den maximal vom User zu belegenden Speicher im Tablespace fest.
PASSWORT EXPIRE – Fordert den User beim ersten Einloggen auf ein neues Passwort einzugeben.
ACCOUNT LOCK—UNLOCK – Der User-Account kann gesperrt oder freigegeben werden.
PROFILE – Weist dem User ein Profil zu.
Beachte: Im CREATE USER-Befehl kann man ein PROFILE aber keine Rolle zuweisen. Einen temporären Tablespace kann man keine Quota für eine User zuweisen.

```
create user name
  identified by password
  default tablespace tbs_1
  default temporary tablespace temp
  quota 10M on tbs_1 qutoa 10M on index_tbs
  password expire
;
```

2.15.3 Externe Authentifikation

Das Betriebssystem übernimmt die Authentifizierung. Die folgende Variable sollte man nicht ändern, da manche Anwendungen diese Voreinstellung verwenden.

```
OS_AUTTHENT_PREFIX =OPS$"
```

```
create user "OPS$domain\user" identified externally;
```

Unix-Shell

```
sqlplus /
```

2.15.4 Externe Passwort-Authentifizierung

Eine Möglichkeit der Identifizierung von SYSDBA und SYSOPER (nicht der Benutzer!) ist die der Nutzung einer sogenannten Passwort-Datei. Diese Passwort-Datei ist bereits standardmäßig angelegt und liegt unter $ORACLE_HOME/dbs. Der Name der Datei lautet pwd*SID*.ora . Bitte löschen Sie diese Datei – auch zu Versuchszwecken – nicht! Wenn Sie die Auswirkungen des Fehlens der Datei testen möchten, verschieben Sie diese an einen anderen Ort.

Um die Nutzung der Passwortdatei einzurichten, ist es notwendig, den Initialisierungsparameter REMOTE_LOGIN_PASSWORDFILE zu ändern. Normalerweise zeigt dieser den Wert NONE an. Für die Nutzung der Passwortdatei muss er den Wert EXCLUSIVE aufweisen. Dieser Parameter ist nicht dynamisch. Ändern Sie den Initialisierungsparameter, indem Sie den Befehl

```
ALTER SYSTEM SET REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE SCOPE=SPFILE;
```

eingeben. Starten Sie Ihre Instanz neu.

Melden Sie sich nunmehr mit dem Privileg SYSDBA an. Verschieben Sie Ihre Passwortdatei pwd*SID*.ora an einen beliebigen anderen Ort (den Sie sich natürlich merken). Melden Sie sich nun von Ihrer Datenbank ab und erneut als SYSDBA an. Ohne Passwortdatei haben Sie nun ein Problem, obwohl Sie über das Privileg SYSDBA verfügen. Eine Meldung, dass Sie nicht über die notwendigen Berechtigungen verfügen verdeutlicht Ihnen die unschöne Situation.

Kopieren Sie die Passwortdatei wieder an Ihren ursprünglichen Ort. Wenn Sie neue Benutzer anlegen und Ihnen das Privileg SYSDBA zuweisen, übernimmt die Passwortdatei die Authentifizierung der Benutzer – aber nur derer, denen das Privileg SYSDBA oder SYSOPER zugewiesen wurde. Gibt es keine Passwortdatei, stehen Ihnen die Privilegien SYSDBA und SYSOPER nicht zur Verfügung und können auch nicht Benutzern zugewiesen werden (unter Privilegien tauchen sie gar nicht auf). Eine Passwortdatei erstellen Sie auf BETRIEBSSYSTEMEBENE - also nicht unter SQL - mit dem Befehl

```
$orapwd file=$ORACLE_HOME/dbs/orapwDB01 password=orapass entries=5
```

Die Passwortdatei muss immer im Verzeichnis $ORACLE_HOME/dbs liegen, um genutzt werden zu können. Der Name ist festgelegt auf pwd*SID*.ora - also pwdtestdb.ora.

Der Parameter password = gibt das Kennwort an, das in der Datei gespeichert wird. Die Wahl des Kennwortes ist beliebig (also z.B. password=geheim). Dieses Passswort wird nicht bei der Anmeldung eingegeben. Es wird nur intern verwendet. Der Parameter entries = legt fest, wie viele Passworteinträge in diese Datei vorgenommen werden können. In unserem Beispiel liegt die Obergrenze bei 20. Mehr SYSDBAs oder SYSOPERs darf es also nicht geben.

Bitte planen Sie Ihre Passwortdatei so, dass alle Benutzer hineinpassen, die als SYSDBA oder SYSOPER fungieren sollen. Eine Erweiterung der Passwortdatei um weitere Einträge ist nämlich nicht möglich. Sollten die möglichen Einträge (entries) nicht ausreichen, müssen Sie eine neue Datei erstellen! Bitte beachten Sie auch, dass dann alle SYSDBA und SYSOPER-Privilegien neu zugewiesen werden müssen. Die bereits bestehenden Privilegien verlieren ihre Gültigkeit. Problematisch stellt sich auch immer die Änderung der Privilegien für den Benutzer SYS dar.

Folgende Reihenfolge ist bei Erstellung einer Passwortdatei zwingend:

1. Erstellen Sie die Passwortdatei
2. Setzen Sie das Initialisierungsparameter
3. Weisen Sie den Benutzern entsprechende Privilegien als SYSDBA bzw. als SYSOPER zu

2.15.5 Einloggen ohne Passwort

SQLPLUS – Unix

Der administrative Datenbankbenutzer, der auf Unix-Ebene Eigentümer aller datanbankrelevanten Dateien ist, darf sich ohne Passwort anmelden.

```
sqlplus "/as sysdba"
```

Diese Berechtigung wird primär über die Zugehörigkeit zur Unix-Gruppe der DBAs geregelt. Der Name dieser Gruppe ist in den meisten Fällen dba. Im Zweifel sieht man sich die Zugehörigkeit der Dateien unterhalb von $ORACLE_HOME an, der Besitzer der Dateien ist fast sicher der administrative Datenbankbenutzer. Die genaue Bezeichnung der DBA-Gruppe findet sich in $ORACLE_HOME/rdbms/lib/config.c unter der Sektion ss_dbs_grp.

```
/*  SS_DBA_GRP defines the UNIX group ID for adminstrative access.  */
/*  Refer to the Installation and User's Guide for further information.  */

#define SS_DBA_GRP "dba"
#define SS_OPER_GRP "dba"

char *ss_dba_grp[] = {SS_DBA_GRP, SS_OPER_GRP};
```

OEM Console – Windows

Wenn Sie Oracle unter Windows betreiben müssen, dann haben Sie (wahrscheinlich nicht nur) ein Sicherheitsproblem. Man kann sich als Windows-Administrator in die OEM Console ohne gültiges Passwort einloggen. Eine Trennung zwischen den Verantwortlichkeiten der System- und Datenbank-Administration ist so kaum möglich.

Lösungsvarianten

1. Windows durch ein ordentliches Server-Betriebssystem ersetzen.
2. Löschen der Windows-Gruppe ORA_DBA

Die erste Variante ist sicherlich der beste Weg. Müssen Sie Windows verwenden, dann bleibt Ihnen nur die Löschung der Windows-Gruppe Oracle_DBA.

```
Active Directory
  Domain
    Users
      ORA_DBA -- hau weg
```

2.15.6 Quotas

Default = 0

Deshalb sollten, ausser auf Undo- und Temporary-Tablespace, Quotas immer angeben werden.

Quota ändern

```
alter user anna quota 0 on tbs_1;
```

Wenn Quota verkleinert wird, werden Daten aber nicht gelöscht.

2.15.7 User löschen

Wenn ein Mitarbeiter die Firma verläßt, nicht den entsprechenden User löschen! Besser: Entziehen Sie dem User das Privileg eine Session zu öffnen.

```
drop user anna cascade;
```

cascade – Alle Objekte des Schemas werden auch gelöscht.

2.15.8 Informationen über User

DBA_USERS

```
describe dba_users;
select username,default_tablespace from dba_users;
```

DBA_TS_QUOTAS

```
select * from DBA_TS_QUOTAS where username ='anna';
```

2.16 Privilegien

Privilegien können nicht bei der Usererstellung in einem Befehl zugewiesen werden.

2.16.1 System-Privilegien

ANY

Systemweit in jedem Schema.

```
grant select any table to anna;
```

Anna kann nur noch ihre eigenen Tabellen abfragen:

```
revoke select any table from anna;
```

WITH ADMIN OPTION

Anna kann diese Privileg an andere User weitergeben.

```
grant select any table to anna with admin option;
```

System-Privilegien sind nicht kascadierend. Beim Entziehen der Berechtigung vom User anna behalten alle User, die System-Berechtigungen vom User anna erhalten haben, diese Rechte.

TO/FROM PUBLIC

```
grant select any table to public;
```

```
revoke select any table from public;
```

SYSDBA / SYSOPER

Spezielle System-Privilegien sind SYSDBA und SYSOPER.

O7_DICTIONARY_ACCESSIBILITY

```
O7_DICTIONARY_ACCESSIBILITY = 'FALSE'
```

```
show parameter o7;
```

2.16.2 Objekt-Privilegien

ON

Der Grant-Befehl muss ein 'ON' haben.

```
grant select ON anna.annatest to berta;
```

WITH GRANT OPTION

Die Weitergabe erfolgt hier mit WITH GRANT OPTION (nicht ADMIN OPTION):

```
grant select on anna.annatest to berta WITH GRANT OPTION;
```

Der Eigentümer des Schemas hat immer die GRANT OPTION auf Objekte seines Schemas.

Entfernen von Objekt-Privilegien

```
revoke select ON anna.annatest from berta;
```

Objekt-Privilegien sind kascadierend. Beim Entziehen der Berechtigung von Anna verlieren auch alle User, die Objekt-Berechtigungen von Anna erhalten haben, diese Rechte.

2.16.3 Informationen über Privilegien

Anzeige der Usern zugewiesenen Systemprivilegien

```
select * from dba_sys_privs where grantee = 'ANNA';
```

Anzeige aller z.Z. verfügbaren Privilegien.

```
select * from session_privs;
```

Anzeige aller Grants von Objekten des Eigentümer anna.

```
select * from dba_tab_privs where owner = 'ANNA';
```

Anzeige von Privilegien von Objekt-Spalten

```
select * from dba_col_privs;
```

2.16.4 Übung Benutzerverwaltung

Übungen siehe Seite 269.

2.17 Rollen

Bei neuen Oracle-Versionen sollen Standard-Rollen entfernt werden. Deshalb keine Standard-Rolle verwenden. Rollen haben keinen Eigentümer. Rollen können nicht bei der Erstellung eines Users diesem in einem Befehl zugewiesen werden.

2.17.1 Rolle erstellen

```
CREATE ROLE role [NOT IDENTIFIED | IDENTIFIED
  {BY password | EXTERNALLY | GLOBALLY | USING package}];
```

NOT IDENTIFIED – Keine Authentifizierung.
IDENTIFIED – Es ist eine Authentifizierung zum Aktivieren der Rolle notwendig.
BY password – Authentifizierung erfolgt mittels Paswort.
EXTERNALLY – Authentifizierung über das Betriebssystem oder einen Third-Party-Service.
GLOBALLY – Authentifizierung über den Enterprise Directory Service.

```
create role oe_bla;
```

2.17.2 Privelegien zuweisen

```
grant select any table to oe_bla;
```

2.17.3 Einem User eine Rolle zuweisen

```
grant role1, role2, role3 to user|role|public with admin option;
```

Max. 148 Rollen können aktiv sein. Standard (MAX_ENABLED_ROLES) sind max. 30 aktive Rollen. Beim Anmelden sind nur die Default-Roles aktiv. Rollen können nicht beim CREATE USER-Befehl zugewiesen werden.

```
alter user scott default role hr_clerk, oe_clerk;
```

Macht alle Rollen, die anna zugewiesen worden, zu Default-Roles, ausser role3.

```
alter user anna default role all except role3;
```

Mit SET ROLE (de-)aktiviert der User für sich dynamisch die Rollen. Man muss sämtliche Rollen beim (De-)Aktivieren angeben. Wird eine vorher aktivierte Rolle bei einem SET-Befehl nicht mit angegeben, wird sie deaktiviert. Der SET-Befehl gilt nur für die Dauer einer Session.

```
set role role2, role3;
set role all except oe_clerk;
```

2.17.4 Kennwort geschützte Rollen

```
set role oe_clerk identified by order;
```

Schützen bestimmte Rollen vor dem Aktivieren. Dies ist z.B. für Web-Anwendungen sinnvoll, bei denen das Passwort in der Anwendung übermittelt wird. Kennwort geschütze Default-Rollen brauchen ein Kennwort.

2.17.5 Rolle entziehen

```
revoke role1 from public;
```

2.17.6 Rolle löschen

Rolle wird allen Usern weggenommen.

```
drop role hr_manager;
```

2.17.7 Rolle zuweisen vs. Rolle aktivieren

- grant – Zuweisen
- set role – aktivieren

2.17.8 Informationen über Rollen

```
select role, password_required from dba_roles;
```

2.18 Auditing

Man kann bis auf Spaltenebene überwachen. Auting ist per default nicht aktiviert.

Wenn nicht mehr überwacht werden kann, kann die Aktion nicht mehr ausgeführt werden. Das heißt wenn die Partition mit Audit-Logs voll ist, läuft nichts mehr.

```
AUTID_TRAIL = DB | OS | FALSE
```

Man kann in eine Oracle-Tabelle schreiben oder besser in eine Betriebssystemlogdatei.

- DB – DB-Tabelle
- OS – Betriebssystem
- FALSE – deaktiviert

2.18.1 Audit in eine Betriebssystem-Log-Datei

```
alter system set AUDIT_TRAIL = OS scope = spfile;
-- NT: Schreibt in die Ereignisanzeige
-- Unix: Angabe des zusätzlichen Parameters: AUTID_FILE_DEST = Verzeichnis
shutdown immediate;
startup;
```

2.18.2 Audit in die Datenbanktabelle SYS.AUD$

```
DD BASE TABLE
SYS.AUD$
Audit trail information stored in
```

Wenn die Ereginisse in die Tabelle SYS.AUD$ gespeichert werden, sollte diese von Zeit zu Zeit mit TRUNCATE geleert werden.

```
alter system set AUDIT_TRAIL = DB scope = spfile;
```

Zum Abfrage dient die View DBA_TRAIL.

```
select username,action_name,to_char(timestamp, 'DD.MM.YYYY HH24:MI:SS'),returncode
  from dba_audit_trail;
```

timestamp zeigt nur das Datum und keine Uhrzeit.

2.18.3 Auditing aktivieren

```
audit create any table;
audit drop any view;
audit drop any table;
audit session by anna;
```

Beim Herunterfahren einer Instanz werden alle Audits deaktiviert.

2.18.4 Auditing deaktivieren

Es ist die entsprechende Syntax wie beim Aktivieren verwenden.

```
noaudit create any table;


select username, action_name, returncode from dba_audit_trail;
select * from aud$;
desc aud$;
```

2.19 Netzwerkbetrieb – Übersicht

2.19.1 Single Network Architecture – 1-Tier

Bei 1-Tier gibt es nur ein Host auf dem die Anwendung läuft.

2.19.2 Simple Network Architecture – 2-Tier

Das einfache Netzwerk verbindet Clients und Server. Beide müssen das gleiche Protokoll verwenden. Steigt die Anzahl von Clients, kann es zu Enpässen kommen und es ist notwendig eine N-Tier Archtektur zu implementieren.

2.19.3 N-Tier

Zwischen Clients und Server wird ein Middle-Tier (Applikationsserver) installiert. Das Middle-Tier kann den Server entlasten. Die Clients können unterschiedliche Protokolle verwenden, da der Applikationsserver dies für den Server in ein Protokoll umsetzt. Das Middle-Tier kann Agent-Dienste für die Clients bereitstellen.

2.19.4 Complex Network Architecture

Typisches Beispiel ist das Internet. Es werden unterschiedliche Betriebssysteme, mit unterschiedlichen Protokollen, weltweit vernetzt. Jede Complex Network Architecture ist N-Tier aber nicht jedes N-Tier ist eine Complex Network Architecture.

2.19.5 Oracle9i Lösungen für den Netzwerkbetrieb

Konnektivität

Oracle Net Services Oracle Net Services hies früher Net-8.

- Protokollunabhängigkeit
- Umfassende Plattformunterstützung
- Integrierte GUI-Verwaltungswerkzeuge
- Vielfältige Konfigurationsoptionen
- Trace- und Diagnose-Werkzeuge
- Basis-Sicherheit

Datenbank-Konnektivität mit IIOP und HTTP: Mit Hilfe eines Webbrowsers kann mittels HTTP und mittels Java-Applet über IIOP (Internet Inter-ORB Protocol) eine Verbindung zur Datenbank aufgebaut werden. Internet-Technologien wie Internet Files System, Enterprise JavaBeans (EJB) und das Standardprotokoll Secure Sockets Layer (SSL) bieten zusätzliche Sicherheit für Netzwerkverbindungen.

Directory Service

Oracle Internet Directory (OID) ist dem LDAP (Version3) kompatibles Directory Service von Oracle.

Skalierbarkeit

Oracle Shared Server: Mit Oracle Shared Server können sich eine große Anzahl von Benutzern gleichzeitig bei einer Datenbank anmelden. Weiteres siehe Seite 158.

Connection Manager: Connection Manager ist ein Gateway-Prozess und Steuerungsprogramm, das auf einer mittleren Netzwerkhierarchie konfiguriert und Installiert ist.

Sicherheit

Advanced Security: Oracle Advanced Security ist ein Extra-Paket von Oracle.

- Verschlüsselung (DES, RSA, 3DES)
- Berechtigungsprüfung (SSL, Kerberos, Radius, CyberCafe)
- Datenintigrität (MD5, SHA)

Firewalls: Oracle unterstützt die Produkte führender Anbieter von Firewalls. Oracle Net Application Proxy Kit ermöglicht es Firewall-Anbietern, Verbindungen zu Oracle Umgebungen zu unterstützen.

Verfügbarkeit

Heterogene Dienste: Heterogene Dienste bieten eine Integration von Oracle Servern und -Umgebungen Nicht-Oracle Servern und Umgebungen.

Externe Prozeduren: Externe Prozeduren sind Funktionen, die unter einer 3GL-Sprache geschrieben sind und so von PL/SQL abgerugen werden können.

2.20 Oracle Net Architektur

Oracle Net ist eine Kommunikationssoftware, welche eine Netzwerkverbindung von einer Client-Applikation zu einem Oracle Datenbankserver ermöglicht. Oracle Net ist eine 'Weiterleitungsschicht', welche über den Netzwerkprotokollen angesiedelt ist und sich sowohl auf dem Client als auch auf dem Server befindet.

Folgende kritische Funktionen werden durch Oracle Net durchgeführt:

- Weiterleitung von Aufrufen (Calls) vom Client zum Server und zurück
- Zeichensatzübersetzung zwischen dem Zeichensatz des Clients und dem Zeichensatz der Datenbank

Aufbau Stack beim Client bzw. Server

- Client Application / RDBMS
- Two-Task Common
- Oracle Net Foundation Layer
- Oracle Protocol Support
- Network Protocol

Client Application / RDBMS

Client Application – Oracle Call Interface (OCI): Die meisten Programmschnittstellen (ADO, ODBC, JDBC etc.) kommunizieren mit Oracle Net über die OCI-Schicht.

Oracle RDBMS: Oracle RDBMS verwendet OPI.

Two-Task Common (TTC): TTC ist verantwortlich für die Anpassung verschiedener Zeichensätze zwischen Client und Server. TTC gehört zur Darstellungsschicht im OSI-Referenzmodell.

Oracle Net Foundation Layer: Oracle Net Foundation Layer ist zuständig für den Aufbau und die Aufrechterhaltung einer Verbindung zwischen Client-Anwendungen und dem Server. Auf der Client-Seite hat der Oracle Net Foundation Layer folgende Aufgaben:

- Auffinden des Standortes des Servers.
- Ermitteln ob ein oder mehrere Protokolle an der Verbindung beteiligt sind.
- Verarbeiten von Ausnahmen und Interrupts.

Auf der Serverseite arbeitet Oracle Net Foundation Layer zusätzlich noch mit dem Listener zusammen. Oracle Net Foundation Layer kommuniziert auch noch mit Namensdiensten und mit Oracle Advanced Security zusammen, um sichere Verbindungen zu gewährleisten.

Oracle Protocol Support (OPS): Oracle Protocol Support stellt die Protokollschnittstelle für verschiedene Netzwerkprotokolle zur Verfügung:

- TCP/IP
- TCP/IP mit SSL
- Named Pipes
- LU6.2 IBM
- Virtual Interface (VI)

2.21 Basiskonfiguration auf der Server-Seite

2.21.1 Der Listener

Der Listener ist ein Prozess, der auf einem Knoten läuft und für eine oder mehrere Datenbank-Instanzen eingehende Verbindungen empfängt.

- Ein Listener kann für mehr als eine Datenbank Verbindungen empfangen.
- Es können mehrere Listener zum Load Balancing für eine Datenbank konfiguriert werden.
- Ein Listener kann für mehrere Protokolle Verbindungen empfangen.
- Der Default-Name ist LISTENER.
- Der Name des Listener muss pro listener.ora eindeutig sein.

Verbindungsmethoden

Wenn ein Client eine Verbindungsanforderung an einen Server stellt, reagiert der Listener mit einer der folgenden Aktionen:

- Starten eines Server-Prozesses (Spawn) und Weitergeben (Bequeath) der Verbindung an ihn.
- Das Handoff der Verbindung an einen Dispatcher in eine Shared Server-Konfiguration.
- Umleiten der Verbindung zu einem Dispatcher oder Server-Prozess (Windoofs).

Dienstekonfiguration und -registrierung

dynamisch

- Braucht keine Konfiguration in `listener.ora`
- Listener benötigt PMON-Prozess

statisch

- bei Oracle8 und früheren Releases
- Benötigt Konfiguration in `listener.ora`
- Erforderlich für Oracle Enterprise Manager und andere Dienste

listener.ora

```
# LISTENER.ORA Network Configuration File:
# /oracle/ora92/network/admin/listener.ora
# Generated by Oracle configuration tools.
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
    )
    (DESCRIPTION =
       (ADDRESS = (PROTOCOL = TCP)(HOST = pinguin)(PORT = 1521))
    )
  )
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /oracle/ora92)
      (PROGRAM = extproc)
    )
      (SID_DESC =
      (GLOBAL_DBNAME = testdb29)
       (ORACLE_HOME = /oracle/ora92)
       (SID_NAME = testdb29)
    )
  )
```

lsnrctl

Mit dem Befehlszeilenprogramm lsnrctl können Sie beliebiege Listener starten und stoppen und eine Reihe weitere Überprüfungsoperationen durchführen. Wenn Sie einen Listener starten wollen so können Sie `start listenername` eingeben. Geben Sie anstelle des Listenernamen nichts ein, wird der Default-Listener namens listener benutzt.

```
lsnrctl stop
lsnrctl start
```

help zeigt alle lsnrctl-Befehle an.

```
lsnrctl
LSNRCTL> help
...
```

Mit SET ist es möglich, Parameterwerte innerhalb des tnslsnr zu ändern.

tnsping

Mit dem Kommandozeilentool `tnsping` kann man die Namensauflösung und das Port testen.

```
tnsping testdb29
```

2.22 Namensauflösung unter Oracle

Damit ein Client sich mit einer Datenbank verbinden kann, muss dieser den Namen der Datenbank auflösen können. Oracle stellt hierfür verschiedene Methoden zur Verfügung:

- Host Naming (HOSTNAME)
- Local Naming (TNSNAMES)
- Oracle Name Server (ONAMES)
- Directory Naming
- Externe Benennung

In kleineren Organisationen wird meist das Local Naming oder Host Naming verwendet. Bei größeren Organisationsformen wird oft Directory Naming mittels LDAP-kompatiblen Directory Server eingesetzt.

2.22.1 sqlnet.ora

Welche der Methoden eingesetzt wird, ermittelt Oracle anhand der Datei `sqlnet.ora`.

```
# SQLNET.ORA Network Configuration File:
#   /oracle/ora92/network/admin/sqlnet.ora
# Generated by Oracle configuration tools.
NAMES.DEFAULT_DOMAIN = testdb29.de
SQLNET.AUTHENTICATION_SERVICES= (NTS)
NAMES.DIRECTORY_PATH = (TNSNAMES, ONAMES, HOSTNAME)
```

Gemäß den Einstellungen für NAMES.DIRECTORY_PATH würde gegenwärtig als erstes die Namensauflösung mit Hilfe von **Local Naming**, anschließend mit Hilfe eines **Oracle Name Servers** und als letztes durch **Hostnaming** durchgeführt werden.

2.22.2 Net Configuration Assistant

Möglichkeit wäre es die Konfigurationsdateien manuell zu editieren. Glücklicherweise stellt Oracle ein Tool zur Verfügung, welches diese Aufgabe übernimmt. Es handelt sich hierbei um den Net Configuration Assistant. Aufruf:

```
$ORACLE_HOME/bin/netca
```

Wenn Sie dieses Tool starten, müssen Sie als erstes auswählen, was Sie machen möchten. Zur Auswahl steht die Konfiguration des Listeners (auf dem Datenbankserver), die Konfiguration von Benennungsmethoden (in der sqlnet.ora), die Konfiguration des lokalen Net Service Names und das Einrichten eines zentralen Name-Servers. Beispiel:

1. Konfiguration von lokalen Net Service Name
2. Hinzufügen
3. Oracle 9i-Datenbank

4. Dienstname: testdb29
5. Netzwerkprotokoll: TCP
6. Host-Name: pinguin
   Standardport (1521)
7. Verbindungstest
   Anmeldenamen: system oder sys
8. Net Service Name: testdb29

2.22.3 Host Naming

Host Naming ermöglicht Benutzern in einer TCP/IP-Umgebung die Auflösung von Host-Namen mit Hilfe bestehender Dienste der Namensauflösung (DNS, /etc/hosts). Clients stellen die Verbindung mittels Host-Name zu einem Oracle Datenbank-Server über die Software Oracle Net Service Client her. Es kann nur eine SID pro Host-Name adressiert werden. Wenn mehrere SID pro Maschine angesprochen werden sollen, müssen jeweils eigene Host-Namen (Alias) für jede SID definiert werden.

Konfiguration auf der Server-Seite

sqlnet.ora : In der `sqlnet.ora` wird HOSTNAME als Bennenungssystem definiert.

```
NAMES.DIRECTORY_PATH= (HOSTNAME, ...)
```

listener.ora : Bei der Namensauflösung durch hostnaming wird davon ausgegangen, dass der globale Datenbankname in der listener.ora dem Hostname des Computers entspricht. Wenn zum Beispiel der Computer, der die Datenbank testdb29 hostet, pinguin heißt, so ist auch der globale Datenbankname pinguin.

```
# LISTENER.ORA Network Configuration File:
#  /oracle/ora92/network/admin/listener.ora
# Generated by Oracle configuration tools.
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
    )
    (DESCRIPTION =
       (ADDRESS = (PROTOCOL = TCP)(HOST = pinguin.tuxdorf.de)(PORT = 1521))
    )
  )
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /oracle/ora92)
      (PROGRAM = extproc)
    )
      (SID_DESC =
      (GLOBAL_DBNAME = pinguin.tuxdorf.de)
```

```
      (ORACLE_HOME = /oracle/ora92)
      (SID_NAME = testdb29)
    )
  )
```

Hinweis: Bei Veränderungen an der listener.ora muß der Listener-Dienst auch neu gestartet werden.

```
lsnrctl stop
lsnrctl start
```

Verbindungsaufbau vom Client

```
sqlplus system/mamanger@pinguin.tuxdorf.de
```

2.22.4 Local Naming

Local Naming ist die einfachste und wohl am meisten verbreitete Methode der Namensauflösung eines Clients.

Konfiguration auf der Server-Seite

sqlnet.ora : In der `sqlnet.ora` wird TNSNAMES als Bennenungssystem definiert.

```
NAMES.DIRECTORY_PATH= (TNSNAMES ...)
```

listener.ora

```
# LISTENER.ORA Network Configuration File:
#  /oracle/ora92/network/admin/listener.ora
# Generated by Oracle configuration tools.
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
    )
    (DESCRIPTION =
       (ADDRESS = (PROTOCOL = TCP)(HOST = pinguin)(PORT = 1521))
    )
  )
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /oracle/ora92)
      (PROGRAM = extproc)
    )
      (SID_DESC =
      (GLOBAL_DBNAME = testdb29)
      (ORACLE_HOME = /oracle/ora92)
      (SID_NAME = testdb29)
    )
```

Konfiguration auf der Client-Seite

tnsnames.ora : Bei der Namensauflösung mit Hilfe von Local Naming wird die Datei `tnsnames.ora`, in der die Verbindungsinformationen (Deskriptoren) stehen, benutzt.

Diese Datei befindet sich standardmäßig im Ordner `$ORACLE_HOME/network/admin`. Über die `tnsnames.ora` kann der Client ermitteln, auf welchem Rechner sich die Datenbank befindet und über welchen Port die Kommunikation durchgeführt werden soll.

```
testdb29.testdb29.de =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = pinguin)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = testdb29)
    )
  )
```

Die Datenbank befindet sich auf einem Server namens pinguin. Sie können hier auch anstelle des Namens eine IP-Adresse eingeben. Die Verbindung zur Datenbank wird über den Port 1521 hergestellt. Über connect_data werden die Verbindungsdaten festgelegt. In unserem Beispiel handelt es sich um eine dedizierte Verbindung und der ServiceName der Datenbank auf dem Server ist testdb29. Der lokale NetServiceName ist in diesem Beispiel testdb29.testdb29.de. Das bedeutet, dass der Client bei der Verbindung dies als ServiceName eingeben muß.

```
sqlplus system/mamanger@testdb29.testdb29.de
```

Die korrekte Namensauflösung können Sie mit dem Programm `tnsping` testen. Hier wird nun überprüft, ob der ServiceName aufgelöst werden kann und das entsprechende Port offen ist.

```
tnsping testdb29.testdb29.de
...
OK
```

2.22.5 Oracle Name Server

Oracle Name Server ist eine zentralisierte Methode der Namensauflösung eines Clients.

sqlnet.ora

Voraussetzung für die Nutzung durch den Client ist natürlich der entsprechenden Eintrag in der sqlnet.ora.

```
NAMES.DIRECTORY_PATH= (ONAMES)
```

Bei der Namensauflösung mit Hilfe von Oracle Name Service wird neben dem NAMES.DIRECTORY_PATH=(ONAMES) auch noch ein bevorzugter Nameserver angegeben. Bei Auflösungsanfragen wird dann dieser Server gefragt.

2.22.6 Directory Naming

Löst den Namen eines Datenbankdienstes oder Net Service in einem Connect-Descriptor auf, der in einem zentralen LDAP-kompatiblen Directory-Server gespeichert ist.

2.22.7 Externe Benennung

Verwendet einen unterstützten Bennenungsdienst Dritter.

2.22.8 Dynamic Service Registration

PMON ist verantwortlich für das dynamische Registrieren von Diensten im Listener. Dynamic Service Registration wird in der Initialisierungsdatei der Datenbank konfiguriert. Eine Konfiguration der listener.ora ist hierfür nicht notwendig. Die Initialisierungsdatei der Datenbank sollte folgende Einträge beinhalten:

```
SERVICE_NAMES
INSTANCE_NAME
```

2.22.9 Übung Namensauflösung

Übungen siehe Seite 271.

2.23 Shared Server

Der Oracle Shared Server wurde aus dem Oracle Multithreaded Server weiterentwickelt. Bei der Multithreaded Server-Konfiguration übernimmt ein Dispatcher-Prozess die Zuordnung von Benutzer-Prozessen und Server-Prozessen. Wird eine neue Benutzer-Verbindung zur Datenbank hergestellt, so werden alle Anweisungen dieser Benutzer-Verbindung durch den Dispatcher-Prozess in eine Anforderungs-Warteschlange (Request-Queue) eingeordnet. Die Server-Prozesse holen nun die Anweisungen aus der Request-Queue heraus und führen diese Anweisungen in der Datenbank aus. Das Ergebnis der Anweisung wird in einer Antwort-Warteschlange (Response-Queue) abgelegt, um vom Dispatcher-Prozess an den Benutzerprozess weitergeleitet zu werden. Es kann somit beispielsweise 100 Benutzerprozesse geben, die von lediglich 10 oder 20 Serverprozessen bedient werden.

Erhält der Listener eine Anforderung von einem Client für eine Shared-Server-Verbindung, so leitet der Listener die Anforderung an den am wenigsten beschäftigten Dispatcher weiter. Dieser kommuniziert nun mit dem Client, um eine Verbindung herzustellen. Diese Vorgehensweise wird als Direct Handoff bezeichnet. Der Client kommuniziert immer mit dem gleichen Dispatcher. Vor Oracle 9i hat der Listener die Adresse des Dispatchers an den Client zurückgegeben. Der Client hat dann die Verbindung initiiert.

Da ein Dispatcher-Prozess nur ein Kommunikationsprotokoll 'bedienen' kann, brauchen Sie für jedes Kommunikationsprotokoll einen Dispatcher-Prozess. Die Server-Prozesse starten beim Starten der Instanz.

2.23.1 Connection Pooling

Wenn alle Dispatchers belegt sind, könnten sich keine neuen User verbinden. Welche Session ist untätig (Idle ca. >3 Sekunden)? Diese Session wird temporär getrennt und muß jetzt selber warten bis eine andere Session Idle wird. Der PMON regelt dies.

Aktivierung des Connection Pooling

```
alter system set
  dispatchers = '(protocol = tcp)
                 (dispatchers = 3)
                 (pool=on)'
;
```

Hinweis:
DISPATCHER ist per Default gleich 0. Dies muss unbedingt erhöht werden, da sonst nur dedizierte Prozesse laufen.

`SHARED_SERVERS`

```
alter system set
  shared_servers = 2
  max_shared_servers = 20
;
```

Empfehlung von Oracle:
SHARED_SERVERS >= 1 (Default = 0).
Default für MAX_SHARED_SERVERS ist 20 oder 2 * SHARED_SERVERS.

Views

```
select * from V$SESSION;
select * from V$DISPATCHER;
select * from V$QUEUE;
select * from V$SHARED_SERVER;
```

2.24 Backup / Recovery

2.24.1 DB Files überprüfen

Eine Datendatei kann mit dem Shell-Tool dbverify überprüft werden. Dieses Shell-Tool ist günstig für Scripte zur Überprüfung des Backups.

```
dbv file=SYSTEM01.DBF
```

Anzeige defekter Datenfiles.

```
select * from v$recover_file;
```

Anzeige benötigter Redo-Logs.

```
select * from v$recovery_log;
```

Anzeige Archive Logs.

```
select * from v$archived_log;
```

2.24.2 OFFLINE-Backup – NOARCHIVELOG

Beim OFFLINE-Backup werden alle notwendigen Dateien mit Hilfe von Betriebssystembefehlen an einen Sicherungsort kopiert. Dieser Sicherungsmechanismus, auch häufig COLD BACKUP genannt, setzt voraus, dass die Datenbank ordnungsgemäß heruntergefahren wurde. Oracle empfiehlt, möglichst jede Woche ein OFFLINE-Backup durchzuführen. Dies ist natürlich nur dann möglich, wenn Ihre Datenbank nicht 24 Stunden an 7 Tagen in der Woche geöffnet sein muss. Für diese Zwecke gibt es eine Reihe weiterer Sicherungsmethoden, die in den folgenden Kapiteln besprochen werden.

Stellen Sie sich vor, Sie haben eine Datenbank namens testdb29, welche sich im NOARCHIVELOG-Modus befindet. Die Control-Dateien haben Sie auf drei Platten verteilt, um im Crash noch eine Control-Datei zur Verfügung zu haben. Sie planen, ein OFFLINE-Backup jede Nacht durchzuführen. Die Sicherung soll in den Ordner /backup01 gespeichert werden. Von dort aus werden dann die darin enthaltenen Dateien auf Band gesichert.

```
Datenbankname: testdb29
Controldatei1     /oracle/oradata/testdb29/control01.ctl
Controldatei2     /control01/testdb29/control02.ctl
Controldatei3     /control02/testdb29/control03.ctl
Datendateien:     /oracle/oradata/testdb29/users01.dbf
                  /oracle/oradata/testdb29/system01.dbf
                  /oracle/oradata/testdb29/temp01.dbf
                  /oracle/oradata/testdb29/rbs01.dbf
                  /oracle/oradata/testdb29/idx01.dbf
Redo-Log-Dateien  /oracle/oradata/testdb29/redo01.log
                  /oracle/oradata/testdb29/redo02.log
                  /oracle/oradata/testdb29/redo03.log
Sicherungsort:   /backup01
```

In diesem Beispiel führen wir die notwendigen Operationen mit Hilfe von SQLPLUS durch.

1. Fahren Sie die Datenbank ordnungsgemäß herunter. Sind noch Benutzer mit der Datenbank verbunden, so wartet SHUTDOWN NORMAL natürlich so lange, bis diese sich abgemeldet haben. Wenn Sie dies vermeiden wollen, so fahren Sie die Datenbank zuerst mit SHUTDOWN IMMEDIATE herunter und anschließend starten Sie die Datenbank mit STARTUP RESTRICT. Jetzt können Sie die Datenbank problemlos wieder mit SHUTDOWN NORMAL herunterfahren. Geben Sie einfach nur SHUTDOWN ein, so wird als Standardwert NORMAL angenommen.

2. Kopieren Sie die Dateien (Datendateien, Controldateien und Redo-Log-Dateien) in den Ordner /backup01.

3. Starten Sie die Datenbank normal mit STARTUP NORMAL.

Genau genommen ist es ausreichend, wenn Sie eine Control-Datei sichern. Ausschlaggebend hierfür ist der Fakt, dass die Controldateien identisch sind und nur aus Fehlertoleranzgründen redundant gehalten werden.

Natürlich ist es für einen Administrator nicht üblich jede Nacht die Sicherung manuell durchzuführen. Aus diesem Grunde wird an dieser Stelle das oben beschriebene Beispiel mit Hilfe eines Skriptes durchgeführt, welches automatisch jede Nacht gegen 22:00 Uhr ausgeführt wird. Als Basis der Automatisierung wird eine Shell-Script-Datei namens `cold-backup.sh` erzeugt.

```
# cold-backup.sh
sqlplus "sys/sys@testdb29 as sysdba" @~/sql-scripte/shutdown.sql
cp /oracle/oradata/testdb29/* /backup01
cp /control01/test/control02.ctl /backup01
cp /control02/test/control03.ctl /backup01
sqlplus "sys/sys@testdb29 as sysdba" @~/sql-scripte/startup.sql
```

Dieses Script ruft SQLPLUS mit der Parameterdatei `shutdown.sql` auf.

```
shutdown immediate
startup restrict
shutdown normal
exit
```

Diese Parameterdatei `shutdown.sql` fährt die Datenbank mit der Option IMMEDIATE herunter und anschließend wird die Datenbank normal mit der Einschränkung RESTRICT gestartet und normal heruntergefahren. Nun wird SQLPLUS durch den Befehl `exit` beendet und die Abarbeitung der Befehle in der `cold-backup.sh` fortgeführt. Es folgen die eigentlichen Copy-Befehle, gefolgt vom abschließenden Aufruf von SQLPLUS mit der Parameterdatei startup.sql.

```
startup
exit
```

Diese Parameterdatei erledigt nichts weiter als das erneute Hochfahren der Datenbank.

2.24.3 OFFLINE-Recovery – NOARCHIVELOG

Bei Beschädigung einer oder mehrerer Datendateien bzw. der gesamten Datenbank kann nur die komplette Datenbank wiederhergestellt werden. Alle Änderungen, die nach der letzten Sicherung durchgeführt wurden, sind nicht wiederherstellbar.

Fall 1

Nehmen wir einmal an, dass sich alle Transaktionen seit dem letzten Cold-Backup noch in den aktuellen Online-Redo-Logs befinden. Dass heißt, alle notwendigen Informationen für das Herstellen des Zustandes zum Zeitpunkt des Beschädigung sind noch im aktuellen Online-Redo-Log vorhanden. Wenn Sie wissen wollen, welche der Online-Redo-Log-Dateien die aktuelle Online-Redo-Log-Datei ist, so finden Sie das über folgende Abfrage heraus:

```
select v1.status,v1.sequence#,v1.first_change#,v2.member
  from v$log v1,v$logfile v2 where v1.group# = v2.group#
;
STATUS  SEQUENCE# FIRST_CHANGE# MEMBER
---------------------------------------------------------------
INACTIVE      232        401561 /oracle/oradata/testdb29/redo01.log
CURRENT       233        401562 /oracle/oradata/testdb29/redo02.log
INACTIVE      231        400998 /oracle/oradata/testdb29/redo03.log
```

In unserem Beispiel ist die Datei redo02.log die aktuelle Online-Redo-Log-Datei.

Wir wollen nun den oben beschriebenen Vorgang testen. Die Datenbank muss nun zuerst einmal durch ein Cold Backup gesichert werden. Nun wird eine neue Tabelle namens big im Tabelspace users erzeugt, ein Datensatz eingefügt und mit COMMIT abgeschlossen. Die Transaktion ist demnach beendet und der Datensatz befindet sich (natürlich erst nach dem Checkpoint) irgendwo in der Datei `/oracle/oradata/testdb29/users01.dbf` (da der Tablespace users nur aus dieser Datei besteht).

```
create table big(s1 int) tablespace users;
insert into big values(1);
commit;
```

Angenommen die Datei users01.dbf wird beschädigt und Sie ersetzen diese durch deren Sicherungsdatei. Die Tabelle big samt Inhalt befindet sich selbstverständlich nicht in dieser Datei, da die Tabelle big zum Zeitpunkt der Sicherung noch nicht existierte. Alle notwendigen Informationen für ein Rollforward befinden sich jedoch im aktuellen Online-Redo-Log (falls kein Log-Switch erfolgte) – und genau das machen wir uns nunmehr zu Nutze.

Als erstes fahren wir die Instanz herunter und kopieren die gesicherte Datei `users01.dbf` nach `/oracle/oradata/testdb29`. Nun fahren Sie die Instanz normal hoch.

```
shutdown abort;
host
cp /backup01/users01.dbf /oracle/oradata/testdb29/
exit
startup;
...
```

```
Datenbank mit Mount angeschlossen.
ORA-01113: Für Datei '3' ist eine Datenträger-Recovery notwendig
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Da die eben wiederhergestellte Datei `users01.dbf` nicht die aktuelle Version der Datei ist, sondern lediglich eine wiederhergestellte Sicherungskopie, ist ein Datenträger-Recovery notwendig. Da sich aber alle notwendigen Informationen im aktuellen Online-Redo-Log befinden, ist dies kein Problem. Die nach der Sicherung erstellte Tabelle ist selbstverständlich wieder da.

```
recover datafilfe '/oracle/oradata/testdb29/users01.dbf';
select * from big;
       S1
---------
        1
```

Fall 2

Berechtigterweise können Sie sich nun die Frage stellen, was passiert, wenn die Veränderungen, die nach dem Cold Backup durchgeführt wurden, sich zwar nicht in der aktuellen Online-Redo-Log-Datei befinden, sondern in einer der anderen Online-Redo-Log-Dateien. Lassen Sie uns diesen Fall daher einmal durchspielen.

Hierfür ist es notwendig, dass Sie als erstes die Tabelle big löschen und danach ein Cold Backup durchführen (für das Cold Backup können Sie am besten das erstellte Skript benutzen). Als nächstes müssen Sie herausfinden, welches Online-Redo-Log die aktuelle Online-Redo-Log-Datei darstellt.

```
select v1.status,v1.sequence#,v1.first_change#,v2.member
  from v$log v1,v$logfile v2 where v1.group# = v2.group#
;
STATUS  SEQUENCE# FIRST_CHANGE# MEMBER
--------------------------------------------------------------------
INACTIVE      232        401561 /oracle/oradata/testdb29/redo01.log
INACTIVE      233        401562 /oracle/oradata/testdb29/redo02.log
CURRENT       234        401564 /oracle/oradata/testdb29/redo03.log
```

Nehmen wir an, dass im Augenblick die Transaktionen in der Online-Redo-Log-Datei `redo03.log` mit der LSN 234 gespeichert werden. Nun erstellen Sie die Tabelle big erneut und führen einen oder zwei Log-Switches durch.

```
create table big(s1 int) tablespace users;
insert into big values(1);
commit;
alter system switch logfile;
alter system switch logfile;
```

Dadurch befindet sich die Transaktion nicht mehr im aktuellen Online-Redo-Log, sondern in einem der beiden anderen Online-Redo-Logs.

```
select v1.status,v1.sequence#,v1.first_change#,v2.member
  from v$log v1,v$logfile v2 where v1.group# = v2.group#
;
STATUS  SEQUENCE# FIRST_CHANGE# MEMBER
--------------------------------------------------------------------
INACTIVE      235        422305 /oracle/oradata/testdb29/redo01.log
CURRENT       236        422306 /oracle/oradata/testdb29/redo02.log
INACTIVE      234        421564 /oracle/oradata/testdb29/redo03.log
```

Nehmen wir an, dassdie LSN 234 noch vorhanden ist. Demnach ist auch noch die Transaktion vorhanden und eine Wiederherstellung sollte möglich sein.

Wie im oberen Beispiel nehmen wir an, die Datei users01.dbf wird beschädigt und Sie ersetzen diese durch deren Sicherungsdatei. Die Tabelle big samt Inhalt befindet sich selbstverständlich nicht in dieser Datei, da die Tabelle big zum Zeitpunkt der Sicherung noch nicht existierte. Alle notwendigen Informationen für ein Rollforward befinden sich jedoch in einer der Online-Redo-Log-Dateien (nicht jedoch in der aktuellen Online-Redo-Log-Datei) und genau das machen wir uns nunmehr zu Nutze.

Als erstes fahren wir die Instanz herunter und kopieren die gesicherte Datei `users01.dbf` nach `/oracle/oradata/testdb29`. Nun fahren Sie die Instanz normal hoch.

```
shutdown abort;
host
cp /backup01/users01.dbf /oracle/oradata/testdb29/
exit
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01113: Für Datei '3' ist eine Datenträger-Recovery notwendig
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Laut Ausgabe ist für die Datei 3 ein Datenträger-Recovery notwendig. Alle notwendigen Transaktionen befinden sich in der Online-Redo-Log-Datei mit der LSN 234, die noch vorhanden ist (`redo03.log`).

```
recover datafilfe '/oracle/oradata/testdb29/users01.dbf';
select * from big;
       S1
---------
        1
```

Ale Ergebnis zeigt sich hier, dass bei Vorhandensein der notwendigen Transaktionsdaten eine Wiederherstellung auch im NOARCHIVELOG möglich ist.

Fall 3

Abschließend steht natürlich die Frage im Raum, was im Falle eines Nichtvorhandenseins der notwendigen Transaktionen passiert. Um diesen Fall zu testen, müssen lediglich mindestens 3 Log-Switches erfolgen. Somit wird bei drei Log-Gruppen die entsprechenden Transaktionen in jedem Fall überschrieben bzw. als überschreibbar markiert. Führen wir dieses Scenario nun kurz durch.

Als initialer Schritt wird wiederum die Tabelle big gelöscht (um den Ursprungszustand wieder herzustellen) und ein Cold-Backup erzeugt. Nun müssen Sie wieder herausfinden, welches Online-Redo-Log die aktuelle Online-Redo-Log-Datei darstellt. Hierfür können wir wieder die oben beschrieben Abfrage benutzen.

```
select v1.status,v1.sequence#,v1.first_change#,v2.member
  from v$log v1,v$logfile v2 where v1.group# = v2.group#
;
STATUS  SEQUENCE# FIRST_CHANGE# MEMBER
----------------------------------------------------------------
```

```
INACTIVE      235        422305 /oracle/oradata/testdb29/redo01.log
INACTIVE      236        422306 /oracle/oradata/testdb29/redo02.log
CURRENT       237        442308 /oracle/oradata/testdb29/redo03.log
```

Angenommen im Augenblick werden die Transaktionen in der Online-Redo-Log-Date `redo03.log` mit der LSN 237 gespeichert. Nun erstellen Sie die Tabelle big erneut, fügen einen Datensatz hinzu und führen anschließend drei Log-Switches durch.

```
create table big(s1 int) tablespace users;
insert into big values(1);
commit;
alter system switch logfile;
alter system switch logfile;
alter system switch logfile;
```

Dadurch befindet sich die Transaktion weder im aktuellen Online-Redo-Log, noch in einem der beiden anderen Online-Redo-Logs. In der unten ausgeführten Abfrage ist zu erkennen, dass die LSN 237 nicht mehr vorhanden ist, die Transaktion also nicht mehr verfügbar ist.

```
select v1.status,v1.sequence#,v1.first_change#,v2.member
  from v$log v1,v$logfile v2 where v1.group# = v2.group#
;
STATUS  SEQUENCE# FIRST_CHANGE# MEMBER
---------------------------------------------------------------
INACTIVE      238        443218 /oracle/oradata/testdb29/redo01.log
INACTIVE      239        443219 /oracle/oradata/testdb29/redo02.log
CURRENT       240        443220 /oracle/oradata/testdb29/redo03.log
```

Wir nehmen nunmehr an, die Datei users01.dbf wird beschädigt und Sie ersetzen diese durch deren Sicherungsdatei. Alle notwendigen Informationen für ein Rollforward befinden sich jedoch nicht mehr in einer der Online-Redo-Log-Dateien.

Als erstes fahren wir die Instanz9 herunter und kopieren die gesicherte Datei `users01.dbf` nach `/oracle/oradata/testdb29`. Nun fahren Sie die Instanz normal hoch.

```
shutdown abort;
host
cp /backup01/users01.dbf /oracle/oradata/testdb29/
exit
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01113: Für Datei '3' ist eine Datenträger-Recovery notwendig
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Wenn wir nun probieren, ein Datenträger-Recovery durchzuführen, so fragt Oracle nach der entsprechenden Datei, in der die Transaktionen drinstehen, in diesem Fall nach der Dateien mit der LSN 237. Da diese jedoch nicht mehr vorhanden ist bzw. eine andere LSN hat, schlägt ein Recovery fehl.

```
recover datafilfe '3';
...
Log angeben: {<RET=suggested | filename | AUTO | Cancel}
```

Die einzige Möglichkeit ist nunmehr ein vollständiges Wiederherstellen der Datenbank unter Verlust der Tabelle big.

Fall 4

In unserem vierten Fall beschäftigen wir uns mit einer ganz neuen Idee. Nehmen wir an, Sie hätten Ihre Datenbank Sonntag 20:00 Uhr mit Hilfe eines Cold Backups gesichert. Weiterhin nehmen wir an, dass Sie im Tablespace users eine sehr wichtige Tabelle mit sehr wichtigen Daten am Montag 10:00 Uhr erstellt haben.

Unglücklicherweise befinden sich diese Änderungen nicht mehr in den Online-Redo-Log-Dateien, so dass eine der weiter oben beschriebenen Methoden hier leider nicht in Frage kommt. Nun wird die Datei `temp01.dbf` beschädigt und die Datenbank muss wiederhergestellt werden.

Da sich die Transaktionen nicht mehr in den Online-Redo-Log-Dateien befinden, ist zuerst einmal von einer kompletten Wiederherstellung auszugehen. Dies würde jedoch einen Verlust der wichtigen Daten vom Montag 10:00 Uhr bedeuten. Daher ist es möglich, hier einen anderen Weg zu wählen. Da sich im Tablespace temp keine wichtigen Daten befinden, kann dieser gelöscht und neu erzeugt werden.

Um dieses Scenario einmal genauer zu betrachten, legen wir wieder die Datenbank testdb29 aus dem vorigen Kapiteln zugrunde. Voraussetzung ist wiederum, dass ein Cold Backup (angenommen Sonntag 22:00 Uhr) dieser Datenbank existiert. Nun (angenommen Montag 10:00 Uhr) erstellen wir die wichtige Datei big und füllen diese mit Datensätzen. Leider wird die Datei `temp01.dbf` im Laufe des Montags beschädigt. Ein erneutes Hochfahren der Instanz scheitert demzufolge. Sie können die beschädigte Datei, die das Öffnen der Datenbank verhindert, mit der Anweisung

```
ALTER DATABASE DATAFILE datafilename OFFLINE DROP;
```

löschen, anschließend die Datenbank öffnen, den Tablespace löschen und danach den Tablespace neu erstellen. Nun können Sie ganz normal weiterarbeiten.

Zusammenfassung

Als Konsequenz des Cold Backup im NOARCHIVE-Log-Modus ergibt sich ein gravierender Nachteil: Es ist nicht möglich ist, alle Daten wiederherzustellen. Es ist im NOARCHIVE-Modus lediglich möglich, die Datenbank bis zum Zeitpunkt der Sicherung wiederherzustellen. Des weiteren ist es für viele Datenbanken nicht möglich, jede Nacht heruntergefahren zu werden. Als Beispiel seien hier nur die zahlreichen Datenbanken im Internet genannt.

Anmerkung: Vergessen Sie nicht, bei einem `shutdown immediate` den angemeldeten Benutzern eine Nachricht zukommen zu lassen, bevor Sie die Datenbank herunterfahren. Sie werden es Ihnen danken.

2.24.4 OFFLINE-Backup – ARCHIVELOG

Bei einer Offline-Sicherung im ArchiveLog-Modus können alle Daten wieder hergestellt werden. In den archivierten Redo-Log-Dateien befinden sich alle notwendigen Transaktionen.

Betrachten wir wieder die Datenbank testdb29. Diese Datenbank sollte sich im ArchiveLog-Modus befinden. Sie müssen die Datenbank hierfür lediglich herunterfahren und folgende Einträge in die init.ora vornehmen:

```
log_archive_start = true
log_archive_dest_1 = "location=/oracle/oradata/testdb29/archive"
log_archive_format = %%ORACLE_SID%%T%TS%S.ARC
```

Durch den ersten Eintrag wird der Hintergrundprozess Archiver, der die Online-Redo-Log-Dateien sichert, automatisch gestartet. Durch den zweiten Eintrag wird das Ziel angegeben, in das die Online-Redo-Log-Dateien gesichert werden sollen. Durch den dritten Eintrag wird lediglich das Format der gesicherten Online-Redo-Log-Dateien definiert.

Anschließend müssen Sie die Datenbank im Status MOUNT versetzen und den Modus in ArchiveLog ändern. Anschließend kann dann die Datenbank geöffnet werden.

```
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

Wir nehmen an, dass diese Datenbank testdb29 durch ein Offline-Backup ganz normal gesichert wurde. Der Sicherungsordner sei /backup01. Hier sollte sich nun eine Sicherung aller Datendateien, Controldateien und Online-Redo-Log-Dateien befinden. Nun erstellen wir eine Tabelle PRODUKT im Tablespace USERS und tragen dort zwei Beispielprodukte ein.

```
select v1.status,v1.sequence#,v1.first_change#,v2.member
  from v$log v1,v$logfile v2 where v1.group# = v2.group#
;
STATUS  SEQUENCE# FIRST_CHANGE# MEMBER
---------------------------------------------------------------
INACTIVE      160        563628 /oracle/oradata/testdb29/redo03.log
CURRENT       161        564809 /oracle/oradata/testdb29/redo02.log
INACTIVE      159        561580 /oracle/oradata/testdb29/redo01.log

create table produkt(nr int, productname char(10)) tablespace users;
insert into produkt values(1,'Milch');
insert into produkt values(2,'Käse');
commit;
alter system switch logfile;
```

Angenommen, dass sich die After-Images beider Produkte (also die Produkte selber) in der Online-Redo-Log-Datei mit der LSN 161 (current) befinden. Nach dem Hinzufügen wurde ein Log-Switch ausgeführt. Das bedeutet, dass die Online-Redo-Log-Datei mit der LSN 161 nicht mehr die aktuelle Online-Redo-Log-Datei ist. Nun fügen wir zwei weitere Produkte hinzu.

```
select v1.status,v1.sequence#,v1.first_change#,v2.member
  from v$log v1,v$logfile v2 where v1.group# = v2.group#
;
STATUS  SEQUENCE# FIRST_CHANGE# MEMBER
---------------------------------------------------------------
INACTIVE      160        563628 /oracle/oradata/testdb29/redo03.log
ACTIVE        161        564809 /oracle/oradata/testdb29/redo02.log
CURRENT       162        565003 /oracle/oradata/testdb29/redo01.log

insert into produkt values(3,'Butter');
insert into produkt values(4,'Kekse');
commit;
alter system switch logfile;
```

Angenommen, dass sich die After-Images dieser beiden Produkte in der Online-Redo-Log-Datei mit der LSN 162 befinden. Die ersten beiden Produkte sind natürlich weiterhin in der Online-Redo-Log-Datei mit der LSN 161. Nun fügen wir zwei weitere Produkte hinzu.

```
select v1.status,v1.sequence#,v1.first_change#,v2.member
  from v$log v1,v$logfile v2 where v1.group# = v2.group#
;
STATUS  SEQUENCE# FIRST_CHANGE# MEMBER
---------------------------------------------------------------
CURRENT       163        565008 /oracle/oradata/testdb29/redo03.log
INACTIVE      161        564809 /oracle/oradata/testdb29/redo02.log
INACTIVE      162        565003 /oracle/oradata/testdb29/redo01.log

insert into produkt values(5,'Waffeln');
insert into produkt values(6,'Brot');
commit;
alter system switch logfile;
```

Angenommen, dass sich die After-Images dieser beiden neuen Produkte in der Online-Redo-Log-Datei mit der LSN 163 befinden. Die ersten Produkte sind natürlich weiterhin in der Online-Redo-Log-Datei mit der LSN 161 bzw. in der Online-Redo-Log-Datei mit der LSN 162. Ausschnitt aus den Online-Redo-Log-Dateien

```
LSN 161   1  Milch
          2  Käse
LSN 162   3  Butter
          4  Kekse
LSN 163   5  Waffeln
          6  Brot
```

Nun fügen wir zwei weitere Produkte hinzu.

```
alter system switch logfile;
select v1.status,v1.sequence#,v1.first_change#,v2.member
  from v$log v1,v$logfile v2 where v1.group# = v2.group#
;
STATUS  SEQUENCE# FIRST_CHANGE# MEMBER
---------------------------------------------------------------
ACTIVE        163        565008 /oracle/oradata/testdb29/redo03.log
CURRENT       164        565013 /oracle/oradata/testdb29/redo02.log
INACTIVE      162        565003 /oracle/oradata/testdb29/redo01.log

insert into produkt values(7,'Honig');
insert into produkt values(8,'Marmelade');
commit;
alter system switch logfile;
```

Angenommen, dass sich die After-Images dieser beiden neuen Produkte in der Online-Redo-Log-Datei mit der LSN 164 befinden. Die ersten beiden Produkte, die sich in der Online-Redo-Log-Datei mit der LSN 161 befinden, sind jedoch nicht mehr verfügbar. Diese Online-Redo-Log-Datei wurde überschrieben. Da sich die Datenbank jedoch im Modus ARCHIVELOG befindet, wurden die nicht aktuellen Online-Redo-Log-Dateien durch den Hintergrundprozess Archiver bereits weggesichert. Sie müssten sich wohlbehalten in genau dem Ordner, den wir in der INIT-Datei als Parameter angegeben haben.

```
cd /oracle/oradata/testdb29/archive
ls -1
TESTT001S00163.ARC
TESTT001S00162.ARC
TESTT001S00161.ARC
```

2.24.5 OFFLINE-Recovery – ARCHIVELOG

Arbeitsschritte:

1. Instanz herunterfahren
2. Alle notwendigen Dateien vom Tape zurück kopieren.
3. `startup mount;`
4. `recover automatic database;`
   Wenn die Archive-Logs woanders liegen
   `recover from '/backup/' database;`

Sollten nun eine oder mehrere Dateien beschädigt sein, so ist eine vollständige Wiederherstellung problemlos möglich. Alle notwendigen Informationen befinden sich in den Online-Redo-Log-Dateien bzw. in den archivierten Redo-Log-Dateien. Wir werden nun den Verlust der Datendatei users01.dbf simulieren.

```
shutdown immediate;
host
rm /oracle/oradata/testdb29/users01.dbf
exit
startup
...
Datenbank mit Mount angeschlossen.
ORA-01157: Datendatei 3 kann nicht identifiziert/gesperrt werden
           Siehe DBWR-Trace-Datei.
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Wir stellen nun die Datendatei users01.dbf von der Sicherung wieder her und versuchen nun erneut, die Datenbank testdb29 zu öffnen.

```
host
cp /backup01/users01.dbf /oracle/oradata/testdb29/
exit
alter database open;
*
FEHLER in Zeile 1:
ORA-01113: Für Datei '3' ist eine Datenträger-Recovery notwendig
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Wir müssen nun die Transaktionen, die seit der Sicherung durchgeführt wurden, erneut durchführen lassen. Dies wird durch ein Recovery erreicht. Da sich in unserem Fall die ganze Recovery-Aktion nur auf eine Datendatei bezieht, muss natürlich auch nur diese eine Datendatei durch ein Recovery wiederhergestellt werden. Für das Recovery der Datendatei users01.dbf benutzen Sie folgende Anweisung:

```
RECOVER DATAFILE '/oracle/oradata/testdb29/users01.dbf';
```

Oracle versucht nun zu ermitteln, welche Redo-Log-Dateien angewendet werden müssen. In unserem Falle sind es die Dateien mit den LSN 161, 162, 163 und 164. Die Redo-Log-Datei mit der LSN 161 befindet sich jedoch nicht mehr unter den Online-Redo-Log-Dateien. Sie wurde jedoch bereits archiviert.

Nun fragt Oracle: **Log angeben**. Hier können Sie entweder mit ENTER den Vorschlag von Oracle annehmen, eine eigene Datei angeben (wenn Sie es wirklich besser wissen), oder die Anweisung AUTO eingeben. AUTO bedeutet, dass Oracle selbständig sich die richtigen archivierten Redo-Log-Dateien heraussucht und die Datendatei wiederherstellt. Wie Sie in dem oberen Beispiel erkennen können, ist im Anschluss alles ordnungsgemäß wieder vorhanden.

2.24.6 ONLINE-Backup

In der Praxis finden Sie eine immer stärker anwachsende Anzahl von Datenbanken, die 24 Stunden am Tag und dies auch noch 7 Tage die Woche Online sein müssen. In diesem Fall ist es natürlich nicht möglich, die Datenbank für ein Cold Backup in regelmäßigen Abständen herunterzufahren. Für diesen Fall gibt es unter Oracle die Möglichkeit, ein **Hot Backup**, auch oft als Online-Backup bezeichnet, auszuführen. Als Voraussetzung hierfür muss sich die Datenbank im Modus ARCHIVELOG befinden.

Hot Backups werden Tablespace-bezogen durchgeführt. Zuerst wird der Tablespace für die Sicherung vorbereitet. Anschließend werden die Dateien des entsprechenden Tablespaces mit Hilfe eines Betriebssystembefehls gesichert und nach dem Sichern der beteiligten Dateien wird die Sicherung.

```
ALTER TABLESPACE users01 BEGIN BACKUP;
host
cp /oracle/oradata/testdb29/users01.dbf /backup01/
exit
ALTER TABLESPACE users01 END BACKUP
```

Sollten Sie versuchen, die entsprechenden Dateien ohne ALTER TABLESPACE BEGIN BACKUP und ALTER TABLESPACE END BACKUP zu sichern, sind diese gesicherten Dateien für eine Wiederherstellung unbrauchbar. Nachdem ein Tablespace mit ALTER TABLESPACE BEGIN BACKUP in den Hot Backup Modus versetzt wurde, ändert sich das interne Verhalten. Die erste gravierende Veränderung bezieht sich auf das Aufzeichnen der Änderungen in den Redo-Log-Dateien. Es wird jetzt nicht mehr lediglich ein After-Image des verändernden Datensatzes aufgezeichnet, sondern jeweils ein Before Image des gesamten Blockes (Oracle-Block – nicht Betriebssystem-Block!), in dem sich der veränderte Datensatz befindet. Desweiteren wird die SCN zu Beginn des Befehls ALTER TABLESPACE BEGIN BACKUP in den Headern der einzelnen Datendateien 'eingefroren', das heißt, bis zum Befehl ALTER TABLESPACE END BACKUP bleibt diese SCN identisch. Folglich befindet sich in der Sicherung der Datendateien in jedem Falle die SCN, die zu Beginn der Sicherung aktuell war. Diese Vorgehensweise ist notwendig, da Oracle nicht weiss, zu welchen Zeitpunkt das Betriebssystem den Block mit dem Header der Datendatei sichert. Würde die SCN während der Sicherung nicht eingefroren werden, so könnte die SCN in den Datendateien der Sicherung größer als die SCN sein, die zu Beginn von ALTER TABLESPACE BEGIN BACKUP aktuell war. Für das korrekte Anwenden der Redo-Log-Einträge ist es jedoch zwingend erforderlich, dass die SCN der gesicherten Datendateien der SCN zu Beginn der Sicherung entsprechen.

Nun kommen wir zur zweiten wichtigen Änderung in der Arbeitsweise. In den Redo-Log-Einträgen werden gesamte Block-Images gespeichert. Warum?

Ein Oracle-Block ist in der Regel ein Vielfaches eines Betriebssystem-Blockes. Die Größe eines Oracle-Blockes wird in der entsprechenden INI-Datei durch den Parameter DB_BLOCK_SIZE (z. B. 8192) festgelegt. Gebräuchliche Werte liegen zwischen 2K und 8k. Wenn das Betriebssystem aber mit einer Blockgröße von 2048 Byte arbeitet, so werden beim Sichern der Datendateien jeweils Blöcke von 2048 Byte gesichert. Ein Oracle-Block wird also nicht als Ganzes zur gleichen Zeit gesichert, sondern die Betriebssystemblöcke können zu verschiedenen Zeiten durch das Betriebssystem gesichert werden. Es ist demnach auch möglich, das in der Sicherung ein Oracle-Block nicht konsistent ist. Nehmen wir hierfür folgendes Beispiel an:

8K entspricht einem Oracle-Block

```
1 Milch    1 DM   2K  1 BS Block
2 Käse     3 DM
3 Butter   5 DM

4 Brot     2 DM   2K  1 BS Block
5 Wurst    7 DM

6 Sekt     8 DM   2K  1 BS Block
7 Äpfel   11 DM

8 Trauben  9 DM   2K  1 BS Block
9 Grütze  12 DM
```

Sie sehen einen Oracle-Block bestehend aus 4 Betriebssystemblöcken. Angenommen gegen 12:00 Uhr soll dieser Block gesichert werden (natürlich im Zuge der Sicherung der gesamten Datendatei bzw. gesamten Datendateien des entsprechenden Tablespaces). Dieser Block gehört zur Datendatei usr02.dbf und diese wiederum zum Tablespace users2. Gegen 12:00 Uhr wird also dieser Tablespace in den Hot Backup Modus versetzt

```
alter tablespace users2 begin backup;
```

Gegen 12:01 wird der Betriebssystemblock gesichert. Dieser Betriebssystemblock sieht dann gesichert folgendermaßen aus:

```
1 Milch   1 DM     2K  1 BS Block
2 Käse    3 DM
3 Butter  5 DM
```

Gegen 12:02 Uhr wird der Preis aller Produkte um 100% erhöht:

```
UPDATE products SET unitprice=unitprice*2;
COMMIT;
```

In diesem Fall werden die entsprechenden Seiten in den Daten-Buffer geladen und entsprechen verändert. Dadurch werden diese Blöcke zu Dirty-Blocks und im Zuge des nächsten Checkpoints auf der Platte aktualisiert. Nehmen wir an, dass ein Checkpoint gegen 12:03 Uhr auftrat und die Dirty-Buffers zurück in die Datendateien geschrieben wurden. Folglich sieht der Oracle-Block 12:03 Uhr folgendermassen aus:

```
1 Milch    2 DM   2K  1 BS Block
2 Käse     6 DM
3 Butter  10 DM

4 Brot     4 DM   2K  1 BS Block
5 Wurst   14 DM

6 Sekt    16 DM   2K  1 BS Block
7 Äpfel   22 DM

8 Trauben 19 DM   2K  1 BS Block
9 Grütze  24 DM
```

Gegen 12:04 Uhr sichert nun das Betriebssystem den zweiten, dritten und vierten Betriebssystemblock. Der gesicherte Oracle-Block würde zum Zeitpunkt 12:05 Uhr nun folgendermassen aussehen:

```
1 Milch    1 DM   2K  1 BS Block
2 Käse     3 DM
3 Butter   5 DM

4 Brot     4 DM   2K  1 BS Block
5 Wurst   14 DM

6 Sekt    16 DM   2K  1 BS Block
7 Äpfel   22 DM

8 Trauben 19 DM   2K  1 BS Block
9 Grütze  24 DM
```

Der Oracle-Block ist in sich nicht mehr konistent. Ein Teil der Preise wurde verändert, ein anderer noch nicht. Hierbei spricht man allgemein von einem Block-Split. Bei einem Recovery würden wir vor einem Problem stehen. Aus genau diesem Grund wird in der Online-Redo-Log-Datei bei einer Veränderung ein Before-Image des Blockes aufgezeichnet, also ein Image mit den alten Preisen.

Wird nun die entsprechende Sicherung (mit teilweise gesplitteten Blöcken) wiederhergestellt, so ist die SCN ja immer noch die, die zum Zeitpunkt des ALTER TABLESPACE BEGIN BACKUP (in unserem Beispiel 100) aktuell war. Da die während der Sicherung veränderten Oracle-Blöcke in den Online-Redo-Log-Dateien stehen, können diese beim Wiederherstellen der Sicherung schnell die inkonsistenten Oracle-Blöcke ersetzen.

Nachdem die Sicherung der entsprechenden Datendateien beendet ist, wird der Hot Backup Modus durch ALTER TABLESPACE END BACKUP wieder aufgehoben. Nachdem der Hot Backup Modus aufgehoben wurde, wird die Checkpoint SCN in den Headern der einzelnen beteiligten Datendateien wieder auf den aktuellen Stand gesetzt und alles geht seinen gewohnten Gang weiter.

Die Konsequenz dieser Arbeitsweise ist natürlich, dass zwischen ALTER TABLESPACE BEGIN BACKUP und ALTER TABLESPACE END BACKUP der Log-Verkehr zunimmt, sprich die Redo-Log-Dateien nehmen an Größe stärker zu als sonst. Folglich sollte man das Hot Backup nur zu Zeiten geringer DML-Aktivität durchführen, also zu Zeiten, wo wenig eingefügt, geändert und gelöscht wird. Desweiteren sollten Sie jeden Tablespace einzeln sichern, um die Zeit, die der Tablespace sich im Hot-Backup-Mode befindet, zu minimieren.

Richtig:

```
ALTER TABLESPACE user BEGIN BACKUP;
 HOST;
 cp /oracle/oradata/testdb29/usr02.dbf /backup01/
 cp /oracle/oradata/testdb29/usr03.dbf /backup01/
 EXIT
ALTER TABLESPACE user END BACKUP;
ALTER TABLESPACE idx BEGIN BACKUP;
 HOST;
 cp /oracle/oradata/testdb29/idx02.dbf /backup01/
 cp /oracle/oradata/testdb29/idx03.dbf /backup01/
 EXIT
ALTER TABLESPACE idx END BACKUP;
```

Falsch:

```
ALTER TABLESPACE user BEGIN BACKUP;
ALTER TABLESPACE idx BEGIN BACKUP;
 HOST;
 cp /oracle/oradata/testdb29/usr02.dbf /backup01/
 cp /oracle/oradata/testdb29/usr03.dbf /backup01/
 cp /oracle/oradata/testdb29/idx02.dbf /backup01/
 cp /oracle/oradata/testdb29/idx03.dbf /backup01/
 EXIT
ALTER TABLESPACE user END BACKUP;
ALTER TABLESPACE idx END BACKUP;
```

Für das Wiederherstellen einer Sicherung benötigen Sie mindestens alle Redo-Log-Dateien, die zwischen ALTER TABLESPACE ... BEGIN BACKUP und ALTER TABLESPACE ... END BACKUP erzeugt wurden. Dies reicht aber dann auch nur für ein Incomplete Recovery. Für ein Complete Recovery benötigen Sie alle Redo-Log-Dateien von Beginn der Sicherung an.

Vorgehensweise beim Hot Backup:

1. Finden Sie heraus, ob sich die Datenbank im ArchiveLog-Modus befindet.

   ```
   ARCHIVE LOG LIST;
   ```

   Falls sich die Datenbank noch nicht im ArchiveLog-Modus befindet, so wechseln Sie in den ArchiveLog-Modus. Hierbei muß die Datenbank gemountet sein:

   ```
   alter database mount:
   alter database archivelog;
   archive log start;
   alter database open;
   ```

2. Da der gesamte Redo-Log-Verkehr für ein späteres Recovery notwendig ist, muß nun die Online-Redo-Log-Datei (LSN) ermittelt werden, in die aktuell geschrieben wird.

   ```
   archive log list;
   ```

   Angenommen, dass die aktuelle Log Sequenz die Nummer 3 ist. Das heisst, die LSN 3 ist die erste LSN, die für das spätere Wiederherstellen erforderlich ist.

3. Nun versetzen Sie den Tablespace in den Hot Backup Mode.

```
ALTER TABLESPACE users BEGIN BACKUP
```

4. Jetzt sichern Sie die zu diesem Tablespace gehörenden Datendateien an den Sicherungsort.

5. Nun deaktivieren Sie für den Tablespace den Hot Backup Mode

   ```
   ALTER TABLESPACE users END BACKUP
   ```

6. Da der gesamte Redo-Log-Verkehr für ein späteres Recovery notwendig ist, muß nun die Online-Redo-Log-Datei (LSN) ermittelt werden, in die aktuell geschrieben wird.

   ```
   archive log list;
   ```

   Angenommen, dass die aktuelle Log Sequenz die Nummer 4 ist. Das heisst, die LSN 4 ist die letzte LSN, die für das spätere Wiederherstellen zwingend erforderlich ist. Für ein Complete Recovery sind natürlich alle Redo-Log-Dateien von LSN 3 beginnend, notwendig.

Anzeige welche Dateien im Backup-Modus sind:

```
select * from v$backup;
```

2.24.7 ONLINE-Recovery

Arbeitsschritte:

1. Instanz herunterfahren
2. Alle notwendigen Dateien vom Tape zurück kopieren.
3. `startup mount;`
4. `recover automatic database;`
   Wenn die Archive-Logs woanders liegen
   `recover from '/backup/' database;`

Die Wiederherstellung eines durch ein Hot-Backup gesicherten Tablespaces soll an einem Beispiel verdeutlicht werden. Nehmen wir an, Sie haben den Tablespace user2 bestehend aus den Dateien usr02.dbf und usr03.dbf mit Hilfe eines Hot Backups Sonntag 22:00 Uhr in den Ordner /backup01 gesichert. Nehmen wir weiterhin an, dass am Montag eine Tabelle namens products erzeugt wurde und in diese Tabelle im Laufe des Montags 1000 Datensätze eingegeben wurden. Diese 1000 eingefügten Datensätze haben zum Beispiel 3 Log Switches verursacht. Am Montag abend ist die Datei usr02.dbf beschädigt und muß wieder hergestellt werden. Um dieses Beispiel zu simulieren, können wir natürlich nicht einfach so 1000 Datensätze einfügen. Wir fügen einfach zu Testzwecken 10 Datensätze ein und machen jeweils nach drei Datensätzen einen manuellen Log-Switch.

1. Als erstes ermitteln wir die erste LSN, die wir benötigen.

   ```
   archive log list;
   ```

   Angenommen, dass die erste LSN, ist die LSN 5.

2. Als zweites führen wir die eigentliche Sicherung durch

```
ALTER TABLESPACE user2 BEGIN BACKUP
host
cp /oracle/oradata/testdb29/usr02.dbf /backup01/
exit
ALTER TABLESPACE user2 END BACKUP
```

3. Nun erstellen wir die Tabelle products im Tablespace user2 und fügen die ersten drei Datensätze ein.

```
create table produkt(nr int, productname char(10)) tablespace users;
insert into produkt values(1,'Milch');
insert into produkt values(2,'Käse');
insert into produkt values(3,'Käse');
commit;
alter system switch logfile;
archive log list;
```

   Angenommen, dass die nächste zu archivierende Log-Sequenz die 6 ist, muß die LSN 5 schon archiviert worden sein. Im Archivierungsorder müsste diese Datei schon zu finden sein (arc00005.001). Die After-Images der drei eben eingefügten Datensätze befinden sich demnach in der Datei arc00005.001.

4. Als viertes werden wir weitere 3 Datensätze einfügen und einen Log-Switch ausführen.

```
insert into produkt values(4,'Brot');
insert into produkt values(5,'Kekse');
insert into produkt values(6,'Mehl');
commit;
alter system switch logfile;
archive log list;
```

   Angenommen, dass die nächste zu archivierende Log-Sequenz die 7 ist, muß die LSN 6 schon archiviert worden sein. Im Archivierungsorder müsste diese Datei schon zu finden sein (arc00006.001). Die After-Images der drei eben eingefügten Datensätze befinden sich demnach in der Datei arc00006.001.

5. Als fünftes werden wir die letzten 4 Datensätze einfügen und einen Log-Switch ausführen.

```
insert into produkt values(7,'Tee');
insert into produkt values(8,'Saft');
insert into produkt values(9,'Nektar');
insert into produkt values(10,'Obst');
commit;
alter system switch logfile;
archive log list;
```

   Angenommen, dass die nächste zu archivierende Log-Sequenz die 8 ist, muß die LSN 7 schon archiviert worden sein. Im Archivierungsorder müsste diese Datei schon zu finden sein (arc00007.001). Die After-Images der drei eben eingefügten Datensätze befinden sich demnach in der Datei arc00007.001.

6. Nun simulieren wir einen Crash der Datei usr2.dbf.

```
shutdown immediate;
host
rm /oracle/oradata/testdb29/usr2.dbf
exit
```

7. Jetzt versuchen Sie, die Instanz wieder hochzufahren. Da die Datei usr02.dbf beschädigt ist, kann aber die Datenbank lediglich gemountet werden.

```
startup;
...
Datenbank mit MOUNT angeschlossen.
ORA-01157: Datendatei 7 kann nicht identifiziert/gesperrt werden
           Siehe DBWR-Trace-Datei.
ORA-01110: Datendatei 7: '/oracle/oradata/testdb29/users02.dbf'
```

8. Nun wird die Datei usr02.dbf aus dem Ordner /backup01 wiederhergestellt. Anschließend versuchen wir, die Datenbank zu öffnen.

```
alter database open;
*
Fehler in Zeile 1:
ORA-01113: Für Datei '7' ist eine Datenträger-Recovery notwendig
ORA-01110: Datendatei 7: '/oracle/oradata/testdb29/users02.dbf'
```

9. Wir müssen also noch ein Datenträger-Recovery durchführen, um die Datenbank wieder wie gewohnt öffnen zu können

```
RECOVER DATABASE testdb29;
ALTER DATABASE OPEN;
```

2.24.8 BACKUP CONTROLFILE

Nachdem Sie eine neue Datendatei zu einem Tablespace hinzugefügt haben, sollten Sie sofort die Control-Datei sichern. Der Grund hierfür ist, dass im Fehlerfall bei Wiederherstellen der veralteten Control-Datei diese neue Datei nicht bekannt wäre.

```
ALTER DATABASE BACKUP CONTROLFILE TO 'control01.bkp';
```

Hierdurch wird die Controldatei in einer Binärdatei namens control01.bkp gesichert.

TO File

Backup von Control-Files

```
alter database backup controlfile to '/backup/control.bkt';
```

Recover

```
shutdown immediate;
-- Control-Dateien kopieren
startup mount;
recover database until cancel using backup control file;
-- Mit CANCEL abbrechen.
alter database open resetlogs;
```

TO TRACE

Das Vorhandensein einer aktuelle Version der Control-Datei ist einer der wichtigsten Faktoren bei einer erfolgreichen Wiederherstellung einer Datenbank. Wie sie wissen, kann man bestimmte Parameter (maxdatafiles, maxlogfiles etc.) nach dem Erstellen einer Datenbank eigentlich nicht mehr ändern. Nur beim Neuerstellen können Sie hierfür neue Werte angeben. Das Wort 'eigentlich' birgt jedoch in sich doch noch eine Möglichkeit, dies bei einer bestehenden Datenbank im Nachhinein zu ändern. Sie müssen lediglich die Control-Datei neu erstellen. Da die Syntax zum Neuerstellen der Control-Datei relativ umständlich ist, können Sie sich die aktuelle Control-Datei skripten lassen. Hierfür führen Sie lediglich folgende Anweisung aus:

```
alter database backup controlfile to trace;
```

Das erstellte Skript finden Sie in dem Ordner, der als USER_DUMP_DEST in der INI-Datei definiert wurde. In unserem Fall liegt die entsprechende Datei in /oracle/admin/testdb29/udump. In diesem Skript finden Sie die entsprechende CREATE CONTROLFILE-Anweisung. Folgende Schritte müssen Sie durchführen, um eine neue Control-Datei zu erstellen:

1. Starten Sie die Datenbank im Modus NOMOUNT
2. Führen Sie die CREATE CONTROLFILE Anweisung aus.
3. Führen Sie gegebenenfalls ein Recover aus.
4. Öffnen Sie die Datenbank.

Oracle empfielt, nach dem Neuerstellen einer Control-Datei eine vollständige Sicherung der Datenbank.

2.24.9 Erstellen einer neuen Datendatei

Sie sollten Sie auch eine neue Daten-Datei sofort sichern, um später im Fehlerfall die Datei wiederherstellen zu können. Nehmen wir an, Sie haben keine Sicherung von einer neu hinzugefügten Daten-Datei. Und wie soll es anders kommen, genau diese Datei geht verloren. Auch für diesen Fall bietet Oracle eine geeignete Methode an. Sie können einfach die Datei neu erstellen. Hierfür bietet Oracle folgenden Befehl:

```
ALTER DATABASE CREATE DATAFILE 'dateiname'
```

Nun müssen Sie nur noch ein Recovery durchführen, um den aktuellen Stand der Datei zu erhalten. Lassen Sie uns ein Beispiel durchspielen. Als Grundlage dient wiederum unsere Datenbank testdb29 mit dem Tablespace user2, bestehend aus den beiden Dateien usr02.dbf und usr03.dbf. Wir werden als erstes eine weitere dritte Datei usr04.dbf hinzufügen.

```
alter tablespace user2
  add datafile '/oracle/oradata/testdb29/usr04.dbf' size 5m;
```

Anschließend werden wir in diesem Tablespace eine Tabelle erzeugen und dort einige Datensätze eintragen und einen Log-Switch erzwingen. Nun werden wir die Instanz herunterfahren, die Datei usr04.dbf löschen und versuchen, die Instanz wieder hochzufahren.

```
create table produkt(nr int, productname char(10)) tablespace users;
insert into produkt values(1,'Milch');
insert into produkt values(2,'Butter');
insert into produkt values(3,'Käse');
commit;
alter system switch logfile;
shutdown immediate;
host;
rm /oracle/oradata/testdb29/usr04.dbf
exit
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01157: Datendatei 9 kann nicht identifiziert/gesperrt werden
           Siehe DBWR-Trace-Datei.
ORA-01110: Datendatei 9: '/oracle/oradata/testdb29/users04.dbf'
```

Wir sehen in diesem Beispiel, dass Oracle natürlich die beschädigte Datei usr04.dbf zum Öffnen benötigt. Hierfür können wir einfach die Datei erstellen und probieren, die Datenbank erneut zu öffnen.

```
alter database create datafile '/oracle/oradata/testdb29/users04.dbf';
alter database open;
*
FEHLER in Zeile 1:
ORA-01113: Für Datei '9' ist eine Datenträger-Recovery notwendig
ORA-01110: Datendatei 9: '/oracle/oradata/testdb29/users01.dbf'
```

Natürlich ist noch ein Datenträger-Recovery für die Datei users04.dbf notwendig. Hierfür müssen alle Redo-Log-Dateien seit Erstellen der beschädigten Datendatei vorhanden sein. Die notwendigen Transaktionen werden dann auf die neu erstellte Datendatei users04.dbf angewendet, so dass diese Datei dann den Stand der beschädigten Datei hat.

```
recover datafile '/oracle/oradata/testdb29/users04.dbf';
alter database open;
```

Woher weiß aber Oracle, wo die notwendigen anzuwendenden Transaktionen für die Datei beginnen? Oracle speichert in der Control-Datei die aktuelle SCN beim Erstellen einer Datei. Ab hier muss dann begonnen werden, die entsprechenden Transaktionen vorwärts zu rollen, um für die Datendatei den aktuellen Stand wiederherzustellen.

Hinweis: Diese Methode funktioniert nur für Datendateien, die erstellt wurden, nachdem die Datenbank in den ArchiveLog-Modus wechselte.

2.24.10 Recover-Möglichkeiten

Mit der Anweisung RECOVER werden Transaktionen auf wiederhergestellte Datendateien, Tablespaces bzw. ganze Datenbanken angewendet.

```
RECOVER DATAFILE datafilename;
RECOVER TABLESPACE tablespacename;
RECOVER DATABASE;
```

Sie können die RECOVER-Anweisung entweder mit SQLPLUS oder im RMAN durchführen.

2.24.11 Incomplete Recovery

Wenn bei der Wiederherstellung einige Daten nicht wiederhergestellt werden, nennt man dies eine unvollständige Wiederherstellung. Diese kann gewollt oder ungewollt auftreten. Löscht jemand um 20:10 Uhr eine wichtige Tabelle, so können Sie ein zeitbasierte Wiederherstellung durchführen. Diese müsste alles bis zum Zeitpunkt 20:09 Uhr wiederherstellen.

Eine ungewollte unvollständige Wiederherstellung könnte durchgeführt werden müssen, wenn Sie bemerken, dass bei der Wiederherstellung eine Redo-Log-Datei fehlt. Prinzipiell gibt es drei verschiedene unvollständige Wiederherstellungsmöglichkeiten

- Zeit-basiert
- SCN-basiert
- Cancel-basiert

Nach dem unvollständigen Wiederherstellen der Datenbank sind die Online-Redo-Log-Dateien natürlich nicht mehr brauchbar. Die Datenbank muß mit

```
ALTER DATABASE OPEN RESETLOGS
```

geöffnet werden. Hierdurch werden die Online-Redo-Log-Dateien zurückgesetzt, die anschließend wieder benutzt werden können.

Zeit-basiert

Nehmen wir an, Sie führen ein Cold-Backup der Datenbank testdb29 am Sonntag 22:00 Uhr durch. Die Dateien werden in den Ordner /backup01 gesichert.

```
shutdown immediate;
host;
cp /oracle/oradata/testdb29/* /backup01
exit
```

Nehmen wir an, Sie fahren die Datenbank am Montag gegen 8:00 Uhr wieder hoch. Folgendes passiert dann in folgenden zeitlichen Abständen:

1. Erzeugen der Tabelle products am Montag um 8:05 Uhr.

   ```
   create table produkt(nr int, productname char(10)) tablespace users;
   ```

2. Einfügen des ersten Datensatzes am Montag um 8:10 Uhr

   ```
   insert into produkt values(1,'Milch');
   commit;
   ```

3. Einfügen des zweiten Datensatzes am Montag um 8:15 Uhr

   ```
   insert into produkt values(2,'Käse');
   commit;
   ```

4. Einfügen des dritten Datensatzes am Montag um 8:20 Uhr

```
insert into produkt values(3,'Butter');
commit;
```

5. Löschen der Tabelle products am Montatg 8:30 Uhr.

   ```
   drop table produkt;
   ```

6. Durchführen eines Log-Switches

   ```
   alter system switch logfile;
   ```

Nun bemerken Sie, dass die gelöschte Tabelle noch gebraucht wird. Sie müssen die Tabelle wieder herstellen. Als erstes stellen Sie die Datenbank von der letzten Sicherung wieder her.

```
shutdown immediate;
host;
cp /backup01/* /oracle/oradata/testdb29/
exit
```

Nun versuchen Sie die Instanz zu erneut hochzufahren und zu öffnen. Es ist ein Recovery notwendig. Wir müssen die Datenbank bis zum Zeitpunkt Montag 8:25 Uhr wiederherstellen. Hierfür führen Sie eine zeitbasierte Wiederherstellung durch

```
RECOVER DATABASE UNTIL TIME 03.02.2003 08:25:00
```

Anschließend öffnen Sie die Datenbank mit der Option RESETLOGS, um die Online-Redo-Logs zurückzusetzen.

```
ALTER DATABASE OPEN RESETLOGS
```

Die Tabelle mit den Produkten ist wieder da, da das Löschen der Tabelle erst nach 8:25:00 geschehen ist.

Cancel-basiert

Eine Cancel-basierte Wiederherstellung muss durchgeführt werden, wenn bei einer Wiederherstellung eine archivierte Redo-Datei fehlt. Es ist dann nunmehr möglich, bis zu dieser Datei wiederherzustellen und anschließend die Datenbank mit der Option RESETLOGS zu öffnen.

Wir wollen nun dieses Beispiel einmal praktisch umsetzen. Voraussetzung hierfür ist, dass sich die Datenbank im ArchiveLog-Modus befindet und der Archiver-Prozess läuft.

```
archive log list;
```

Im ersten Schritt führen wir eine vollständige Sicherung als Cold Backup durch

```
shutdown immediate;
host;
cp /oracle/oradata/testdb29/* /backup01/
exit
```

Im zweiten Schritt starten fahren wir die Instanz wieder hoch und schauen uns das aktuelle Log an. Wir erzeugen dann eine Tabelle mitarbeiter im Tablespace users und fügen dort zwei Datensätze ein. Diese Datensätz stehen demnach im Log Nummer 9.

```
startup;
archive log list;
create table mitarbeiter(MiNr int, MiName varchar2(30)) tablespace users;
insert into mitarbeiter values(1,'Anna');
insert into mitarbeiter values(2,'Berta');
commit;
```

Jetzt führen wir zwei Log-Switches durch.

```
alter system switch logfile;
alter system switch logfile;
archive log list;
```

Durch den Archiver-Prozess wurden folgende Logs gesichert.

```
ls -1
ARC00007.001
ARC00008.001
ARC00009.001
ARC00010.001
```

Wir fügen zwei Datensätze ein. Diese Datensätz stehen demnach im Log Nummer 11.

```
archive log list;
insert into mitarbeiter values(3,'Carla');
insert into mitarbeiter values(4,'Doris');
commit;
```

Jetzt führen wir zwei Log-Switches durch.

```
alter system switch logfile;
alter system switch logfile;
archive log list;
```

Durch den Archiver-Prozess wurden folgende Logs gesichert.

```
ARC00007.001
ARC00008.001
ARC00009.001
ARC00010.001
ARC00011.001
ARC00012.001
```

Wir fügen zwei wietere Datensätze ein. Diese Datensätz stehen demnach im Log Nummer 13.

```
archive log list;
insert into mitarbeiter values(5,'Emil');
insert into mitarbeiter values(6,'Frank');
commit;
```

Jetzt führen wir zwei Log-Switches durch.

```
alter system switch logfile;
alter system switch logfile;
archive log list;
```

Zusammengefasst stehen folgende Redo-Einträge in folgenden archivierten Logs.

```
Log Nummer  9   1 Anna  und 2 Berta
Log Nummer 11   3 Carla und 4 Doris
Log Nummer 13   5 Emil  und 6 Frank
```

Jetzt wird die Datei users01.dbf des Tablespace users beschädigt. Es muss eine Wiederherstellung erfolgen. Zu allem Schaden ist auch noch das archivierte Log arc00011.001 (mit der Carla und der Doris) verloren gegangen. Die Datenbank fährt nicht mehr ordnungsgemäß hoch.

```
shutdown immediate;
host;
rm /oracel/ora92/rdbms/ARC00011.001
rm /oracle/oradata/testdb29/users01.dbf
exit
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01157: Datendatei 7 kann nicht identifiziert/gesperrt werden
           Siehe DBWR-Trace-Datei.
ORA-01110: Datendatei 7: '/oracle/oradata/testdb29/users01.dbf'
```

Wir versuchen nun, die Datei users01.dbf von der Offline-Sicherung wiederherzustellen und ein Recover durchzuführen. Bedenken Sie, dass in der users01.dbf zum Zeitpunkt der Offline-Sicherung die Tabelle mitarbeiter noch nicht existiert.

```
host;
cp /backup01/* /oracle/oradata/testdb29/
exit
recover datafile 7;
...
ORA-00289: Vorschlag: /oracel/ora92/rdbms/ARC00009.001
Log angeben: {<RET=suggested | filename | AUTO | Cancel}
auto
Wiederherstellung nicht mehr erforderlich.
...
ORA-00308: Archive-Log '/oracel/ora92/rdbms/ARC00011.001'
  kann nicht geöffnet werden.
```

Oracle hat hier nach den archivierten Log-Dateien gefragt, die anzuwenden sind. Es wurde hierbei AUTO angegeben, um Oracle zu veranlassen, die archivierten Logs selbständig zu ermitteln und anzuwenden. Leider fehlt das archivierte Log arc00011.001 (mit unserer Carla und unserer Doris).

Versuchen wir ein Cancel-basiertes Recovery der Datenbank mit der Offline-Sicherung der Datei users01.dbf. Wir geben hier nicht AUTO ein, sondern bestätigen jedes archivierte LOG mit ENTER bis zum achivierten Log arc00011.001. Dort geben wir CANCEL ein, um die Sicherung anzubrechen.

```
host;
cp /backup01/* /oracle/oradata/testdb29/
exit
recover database until cancel;
...
Log angeben: {<RET=suggested | filename | AUTO | Cancel}
cancel
```

Wie wir leider feststellen müssen war das Recovery zwar erfolgreich, ein Öffnen der Datenbank mit der Option RESETLOGS zum Zurücksetzen der Online-Redo-Log-Dateien würde fehlschlagen, da die anderen Backup-Dateien nicht 'alt' genug sind.

```
alter database open resetlogs;
*
FEHLER in Zeile 1:
ORA-01152: Backup-Datei zum Wiederherstellen der Datei 1
  war nicht alt genug.
```

Die Ironie des Schicksals ist es nun, die gesamte Datenbank von der letzten Sicherung wiederherzustellen und anschließend das Cancel-basierte Recovery durchzuführen. Hiermit verlieren wir leider nicht nur den Datensatz von Carla und Doris, sondern auch alle anderen Datesätze, die nach dem LOG 10 in andere Tabellen anderer Tablespaces eingegeben wurden und deren Dateien womöglich noch in Ordnung sind.

2.24.12 Übung Cold Backup im Noarchive-Log-Modus

Übungen siehe Seite 273.

2.24.13 Übung Cold Backup im Archive-Log-Modus

Übungen siehe Seite 274.

2.24.14 Übung Control File skripten / wiederherstellen

Übungen siehe Seite 277.

2.25 Recovery Manager (RMAN)

Oracle liefert den Recovery Manager alsein leistungsstarkes und mächtiges Utility zur Sicherung und Wiederherstellung von Datenbanken. Hierbei speichert der Recovery Manager die durchgeführten Sicherungs- und Wiederherstellungsoperationen. Diese werden entweder in der Control-Datei oder in einem sogenannten Recovery-Catalog gespeichert.

Die Sicherung erfolgt in einem nur vom Recovery Manager lesbaren Format und erfolgt in einer Datei oder auf Band. Für die Sicherung auf Band muss ein Medien-Manager installiert sein (z.B. der Legato Storage Manager). Nur ein Benutzer mit SYSDBA-Priviliegien kann Sicherungs- und Wiederherstellungsoperationen durchführen.

Inkrementelle Sicherung

Ein inkrementelle Sicherung basiert auf einer Ebenen-Struktur. Es werden nur die Blöcke gesichert, die sich seit der letzten Sicherung der gleichen Ebene oder einer tieferen Ebene verändert haben.

- Level 0
  Vollsicherung
- Level 1
  Alle Blöcke, die seit der letzten Level 1- oder Level 0-Sicherung verändert wurden.
- Level 2
  Alle Blöcke, die seit der letzten Level 2-, Level 1- oder Level 0-Sicherung verändert wurden.
- Level 3
  Alle Blöcke, die seit der letzten Level 3-, Level 2-, Level1- oder Level 0-Sicherung verändert wurden.

Schauen wir uns das Ganze an einem Beispiel etwas genauer an:

Sie haben eine Level 0-Sicherung durchgeführt.
Sie führen monatlich eine Level 1-Sicherung durch.
Sie führen wöchentlich eine Level 2-Sicherung durch.
Sie führen täglich eine Level 3-Sicherung durch.

Angenommen, es tritt ein Datenbankfehler genau nach 3 Monaten, 2 Wochen und 3 Tagen auf. In diesem Fall muß die Level 0-Sicherung zurückgesichert werden. Im Anschluß müssen die 3 monatlich durchgeführten Level 1-Sicherungen wiederhergestellt werden. Nun müssen noch die 2 wöchentlich durchgeführten Level 2-Sicherungen durchgeführt werden. Zuletzt müssen nun die 3 täglich durchgeführten Level 3-Sicherung wiederherstellen.

2.25.1 CONFIGURE / REPORT / LIST / SHOW

Default

```
RMAN> configure default device type disk format '/db01/BACKUP/\$U';
```

disk – Festplatte
sbt – Tape
Per Default wird in $ORACLE_HOME/database gespeichert.

Dauerhafte Konfigurationseinstellungen anzeigen

```
RMAN> show all;
```

Auflisten der Backups aller Dateien in der Datenbank

```
RMAN> list backup of database;
```

Alle Backup Sets mit der Datendatei users01.dbf anzeigen

```
RMAN> list backup of datafile "/db01/ORADATA/u03/users01.dbf";
```

Alle Kopien der Datendateien im Tablespace SYSTEM auflisten.

```
RMAN> list backup of tablespace "SYSTEM";
```

REPORT – Erstellt eine detaillierte Analyse des Repository

Anzeige der Datenbankstruktur

```
RMAN> report schema
```

Für welche Dateien ist ein Backup erforderlich?

```
RMAN> report need backup;
```

Welche Backup sind veraltet und können gelöscht werden?

```
RMAN> report obsolete;
```

2.25.2 Sicherung mit Control-Datei

Neben der Möglichkeit, RMAN-Daten in einem Recovery-Catalog zu speichern, können Sie auch RMAN-Daten in den Control-Dateien speichern. Hierbei ist es von enormer Bedeutung, dass Sie Ihre Control-Dateien über mehrere Platten verteilt haben. Es muss immer eine aktuelle Control-Datei vorhanden sein. Falls alle Control-Dateien verloren gegangen sein sollten, und Sie trotzdem die Datenbank wiederherstellen müssen, so können Sie auch von einer Sicherung der Control-Datei die Wiederherstellung per RMAN initiieren.

In unserem Beispiel gehen wir davon aus, dass die Datenbank testdb29 mit dem RMAN gesichert werden muss. Als erstes starten wir hierfür den RMAN.

```
rman target sys/sys@testdb29 nocatalog
RMAN>
```

Nun wird die eigentliche Sicherung der Datenbank durchgeführt. Hierfür wird als erstes ein Kanal zugewiesen.

```
ALLOCATE CHANNEL channelname
         TYPE medientyp_des_sicherungsmediums
         FORMAT Bennenungskonvention_der_Sicherungsdatei;
```

Wir wollen in unserem Beispiel in eine Datei des Ordners /backup01 sichern. Der Name der Sicherungsdatei sollte folgende Variablen enthalten:

%u – 8-Zeichen-String (Nummer des Sicherungsatzes, Erstellungsdatum).
%s – Nummer des Sicherungssatzes.
%p – Teilnummer des Sicherungssatzes (beginnt bei 1).

Wir benutzen demnach folgende Anweisung für die Zuweisung eines Kanals für die Sicherung:

```
ALLOCATE CHANNEL s1
         TYPE disk
         FORMAT '/backup01/b_%u_%s_%p';
```

Anschließend erfolgt die eigentliche Sicherung.

```
BACKUP DATABASE;
```

Diese beiden Anweisungen werden nun noch durch ein RUN zusammengefasst.

```
run {
  ALLOCATE CHANNEL s1
           TYPE disk
           FORMAT '/backup01/b_%u_%s_%p';
  BACKUP DATABASE;
}
```

Recovery

Für das Wiederherstellen mit dem RMAN ist es wichtig, über eine Control-Datei zu verfügen. Als erstes erstellen wir eine kleine Beispieltabelle, um nach dem Wiederherstellen sehen zu können, ob tatsächlich alles bis zum Crash wiederhergestellt werden konnte.

```
create table obst(nr number(3), obstname varchar2(10))
  tablespace users;
insert into obst values(1,'Erdbeeren');
insert into obst values(2,'Kirschen');
insert into obst values(3,'Himbeeren');
commit;
```

Nun fahren wir die Instanz herunter und löschen die Datendatei users01.dbf, um einen Crash zu simulieren.

```
shutdown immediate;
host;
rm /oracle/oradata/testdb29/users01.dbf
```

Anschließend versuchen wir, die Instanz wieder hochzufahren.

```
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01157: Datendatei 3 kann nicht identifiziert/gesperrt werden
           Siehe DBWR-Trace-Datei.
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Wir müssen nun die Datendatei mit RMAN wiederherstellen.

1. Im ersten Teil muss wiederum ein Kanal zugeordnet werden:

   ```
   ALLOCATE CHANNEL s1 TYPE disk FORMAT '/backup01/b_%u_%s_%p';
   ```

2. Im zweiten Teil muss die Datendatei wiederhergestellt werden:

   ```
   RESTORE DATAFILE /oracle/oradata/testdb29/users01.dbf;
   ```

3. Im dritten Teil müssen alle Transaktionen, die seit der Sicherung durch RMAN durchgeführt wurden, auf diese Datendatei angewendet werden.

   ```
   RECOVER DATAFILE /oracle/oradata/test/users01.dbf;
   ```

Diese Teile müssen nunmehr mit RUN zusammengefasst und mit dem RMAN ausgeführt werden.

```
RUN {
  ALLOCATE CHANNEL s1 TYPE disk FORMAT '/backup01/b_%u_%s_%p';
  RESTORE DATAFILE '/oracle/oradata/test/users01.dbf';
  RECOVER DATAFILE '/oracle/oradata/test/users01.dbf';
}
```

Nun versuchen wir, die Datenbank zu öffnen und schauen, ob unser Obst wieder da ist.

```
alter database open;
select * from obst;
 NR  OBSTNAME
--------------
  1  Erdbeeren
  2  Kirschen
  3  Himbeeren
```

Automatische Sicherung der Control-Datei

Sie können die Control-Datei mit Hilfe von RMAN automatisch sichern lassen.

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON
```

In folgenden Fällen wird dann die Control-Datei gesichert.

1. Nach jeder Copy oder Backup-Anweisung an der RMAN-Eingabeaufforderung.
2. Immer wenn innerhalb eines RUN-Blockes einem Copy- oder Backup-Befehl eine anderen Anweisung folgt.
3. Am Ende eines Run-Blockes, wenn die letzte Anweisung ein Copy oder Backup-Anweisung ist.
4. Bei strukturellen Änderungen an der Datenbank.

2.25.3 Sicherung mit Recovery-Catalog

Der Recovery-Catalog speichert alle Sicherungs- und Wiederherstellungsoperationen und erlaubt neben der Wiederherstellung eine Reihe von nützlichen Analysefunktionen. Der Recovery-Catalog benötigt einen eigenen Tablespace. Diesen Tablespace in der zu sichernden Datenbank zu erstellen wäre ein Fehler. Im Falle des Crashs der Datenbank wäre auch der Recovery-Catalog nicht mehr ansprechbar, und damit wäre eine Wiederherstellung nicht mehr möglich. Empfehlenswert ist das Erstellen einer eigenen Datenbank für den Recovery-Catalog und das Sichern dieser Recovery-Catalog-Datenbank mit Hilfe eines Cold- oder Hot-Backups.

Recovery-Catalog erstellen

Voraussetzung für das Erstellen eines Recovery-Cataloges ist ein eigener Tablespace. Desweiteren brauchen wir einen Benutzer, der Besitzer dieses Recovery-Cataloges ist. Hierfür muss sich diesem Benutzer die Rolle RECOVERY_CATALOG_OWNER, CONNECT und RESOURCE zugewiesen werden.

1. Erstellen der Catalog-Datenbank (in unserem Beispiel heißt diese Datenbank cat). Sie können hierfür den Database Configuration Assistent benutzen.

2. Neuen Tablespace erstellen

```
create tablespace rman_ts
   datafile '/rman-katalog/rman_ts01.dbf' size 20M
   default storage (initial 100K next 100K pctincrease 0)
;
```

3. Erstellen des RMAN-Users (in unserem Beispiel heißt dieser Benutzer rman).

```
create user rman identified by rman
  default tablespace users
  temporary tablespace temp
;
grant connect to rman;
grant ressource to rman;
grant recovery_catalog_owner to rman;
```

4. Der entsprechende Catalog wird nun erstellt. Dies muss mit Hilfe des Recovery Manager RMAN geschehen. Hierfür müssen Sie den Recovery Manager starten und sich mit der entsprechenden Catalog-Datenbank verbinden.

```
rman catalog rman/rman@cat
RMAN> create catalog tablespace 'users';
```

5. Die zu sichernde Datenbank oder Datenbanken müssen nun registriert werden. Hierfür starten Sie wieder der Recovery Manager RMAN unter Angabe der Catalog-Datenbank und unter Angabe der Zieldatenbank.

```
rman catalog rman/rman@cat target internal/oracle@testdb29
RMAN> register database;
```

Die Sicherung selbst erfolgt wie die Sicherung einer Control-Datei.

Die Sicherung mit einem Recovery-Catalog bietet darüber hinaus eine Reihe von nützlichen Erweiterung. So können Sie im Recovery-Catalog Skripte speichern, welche die Sicherung einer Datenbank oder Teile der Datenbank (Tablespaces, Dateien etc.) automatisieren. Folgendes Skript führt ein Cold Backup der Datenbank testdb29 durch:

```
replace script 'Vollsicherung' {
  shutdown immediate;
  startup mount;
  allocate channel ch1 type disk format '/backup/%d_DB_%u_%s_%p';
  backup database include current controlfile;
  release channel ch1;
  alter database open;
}
```

Aufgerufen wird das Skript folgendermaßen:

```
run {execute script Vollsicherung;}
```

Das Listing des Skriptes können Sie sich so anschauen:

```
print script "Vollsicherung";
```

Berichts- und Überprüfungsfunktionen

Anzeigen aller Sicherungsdetails.

```
list backup;
```

Anzeigen bzw. Löschen von Sicherungen, die nicht mehr für das Wiederherstellen benötigt werden.

```
report obsolete;
delete obsolete;
```

Anzeigen bzw. Löschen von Sicherungen, die nicht mehr für das Point In Time-Wiederherstellen innerhalb der letzten sieben Tage benötigt werden.

```
report obsolete recovery window of 7 days;
delete obsolete recovery window of 7 days;
```

Anzeigen bzw. Löschen von Sicherungen, bei denen zwei neuere Kopien zur Verfügung stehen.

```
report obsolete redundancy = 2 device type disk;
delete obsolete redundancy = 2 device type disk;
```

Anzeigen von Dateien, die nicht wiederhergestellt werden können.

```
report unrecoverable database;
```

Synchronisierung des Kataloges mit den Control-Dateien.

```
resync catalog
```

Überprüfen, ob die Sicherungsdateien existieren.

```
crosscheck backup;
```

Ein Cross-Check markiert die Sicherungen, die gemäß der Retention Policy noch benötigt werden, entweder als AVAILABLE (falls die Sicherungen vefügbar sind) oder als EXPIRED (wenn die Sicherungen nicht verfügbar sind) Mit dem folgenden Befehl können Sie sich die Sicherungen, die nicht verfügbar sind, anzeigen lassen.

```
LIST EXPIRED;
```

2.25.4 Umbenennen einer Datei bei der Wiederherstellung

Häufig tritt der Fall ein, dass eine Wiederherstellung aufgrund eines Plattenausfalls notwenig wird. Hierbei ist zu beachten, dass bei einer normalen Wiederherstellung die Dateien immer an ihrem Originalort wiederhergestellt werden. Wird die fehlende Platte jedoch nicht ersetzt, so muß die dort gespeicherte Datei oder Dateien an einem anderen Ort wiederhergestellt werden.

```
SET NEWNAME FOR DATAFILE x TO location;
```

Schauen wir uns das an einem kleinen Beispiel etwas genauer an. Wir gehen von einer Datenbank aus, bei der sich die Datendatei users01.dbf auf /data_01/users01.dbf befindet. Diese Platte fällt nunmehr aus und Sie möchten die Datendatei users01.dbf auf der /data_02/ wiederherstellen.

```
run {
  SET NEWNAME FOR DATAFILE '/data_01/users01.dbf ' TO '/data_02/users01.dbf ';
  RESTORE DATAFILE x;
  RECOVER DATAFILE x;
}
```

Durch set newname wird für die datei /data_01/users01.dbf der neue Speicherort /data_02/users01.dbf festgelegt. Anschließend wird diese Datei wiederhergestellt (schon an dem neuen Speicherort) und alle notwendigen Transaktionen angewendet. Danach kann die Datenbank wieder normal geöffnet werden.

2.25.5 Parallelisieren von Kanälen

Kanäle können aus Geschwindigkeitsgründen parallele Operationen durchführen. Schauen wir uns hierfür als erstes folgendes Beispiel an:

```
run {
  allocate channel c1 type disk;
  allocate channel c2 type disk;
  allocate channel c3 type disk;
  backup datafile 1;
  backup datafile 2;
  backup datafile 3;
}
```

In diesem Beispiel werden zwar drei Kanäle zugeordent, es erfolgen jedoch drei unabhängige Befehle für die Sicherung

```
backup datafile 1;
backup datafile 2;
backup datafile 3;
```

Oracle arbeitet erst eine Anweisung komplett ab, bevor weitere Anweisungen durchgeführt werden. Folglich ist immer nur ein Kanal zu aktiv. Versuchen wir, das ganze etwas umzuschreiben:

```
run {
  allocate channel c1 type disk;
  allocate channel c2 type disk;
  allocate channel c3 type disk;
  backup datafile 1, 2, 3;
}
```

In diesem neuen Skript führt eine Sicherungsanweisung die Sicherung für die drei Datendateien durch (BACKUP DATAFILE 1, 2, 3). Folglich sind alle drei Kanäle aktiv und können genutzt werden.

Ähnlich verhält es sich bei der Sicherung einer gesamten Datenbank. Nehmen wir an, dass eine Datenbank aus 16 Dateien besteht. Sie führen nun folgendes kleine Skript zur Sicherung der Datenbank aus:

```
run {
  allocate channel c1 type disk;
  allocate channel c2 type disk;
  allocate channel c3 type disk;
  backup database;
}
```

Auch in diesem Beispiel kann nur ein Kanal genutzt werden, da die Datenbank als Ganzes in eine Datei gesichert wird. Schreiben wir nun das Ganze etwas um:

```
run {
  allocate channel c1 type disk;
  allocate channel c2 type disk;
  allocate channel c3 type disk;
  backup database filesperset 4;
}
```

Da in ein Sicherungsset nur vier Dateien eingeschlossen werden können (FILESPERSET=4), erfolgt die Sicherung in vier Sicherungssets (16 Dateien - >vier Dateien je Set - >4 Sets). In diesem Fall können alle Kanäle benutzt werden, da in vier Sets gesichert wird.

2.25.6 Automatische Kanalzuweisung

Sie können den Standardgrad bezugnehmend auf die parallele Ausführung angeben.

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 4;
```

Der Parallelitätsgrad wird hierdurch auf vier festgelegt. Das bedeutet, dass im Zuge einer Sicherung auch vier Kanäle genutzt werden können. Nehmen wir an, Sie führen folgende Anweisung zur Sicherung Ihrer Datendateien 1,2 und 3 aus:

```
run {
  backup datafile 1, 2, 3;
}
```

In diesem Fall werden drei Kanäle zugeordnet und für die Sicherung benutzt.

2.25.7 Trial Recovery

Mit Hilfe eines Trial Recovery ist es möglich, eine Wiederherstellung von Redo-Logs 'zu simulieren'. Trial Recovery kommt zum Einsatz, wenn Logs, die wiederhergestellt werden müssen, korrupte Blöcke beinhalten. Die korrupten Blöcke können in der Alert-Log-Datei eingesehen werden. Nach Beendigung der Wiederherstellung werden alle Änderungen rückgängig gemacht. Die korrupten Blöcke werden im Speicher markiert.

Nachdem der Administrator nun weiß, wie viel und welche Blöcke der Redo-Log-Dateien korrupt sind, kann er entscheiden, ob er die Wiederherstellung an diesem Punkt abbricht und die Datenbank mit OPEN RESETLOGS öffnet oder ob er die Wiederherstellung weiterführt und mit ALLOW n CORRUPTION eine gewisse Anzahl korrupter Blöcke 'zulässt'.

2.25.8 FAST_START_MTTR_TARGET

Durch FAST_START_MTTR_TARGET können Sie festlegen, wie lange das Crash-Recovery maximal dauern soll. Angenommen, Sie setzen FAST_START_MTTR_TARGET auf 180, so bedeutet diese Einstellung, dass die Instanz nach einem Fehler (Stromausfall etc.) nach maximal 180 Sekunden, also 3 Minuten, wieder verfügbar ist.

Durch die Einstellung FAST_START_MTTR_TARGET wird also bestimmt, wann Dirty Blocks vom Cache in die Datendateien geschrieben werden. Es wird das Checkpointverhalten verändert und gemäß den getroffenen Einstellungen angepasst.

Folgende Optionen sind in diesem Fall überflüssig:

LOG_CHECKPOINT_INTERVAL
Gibt an, wie oft ein Checkpoint ausgelöst wird, basierend auf der Anzahl der in die Logs geschriebenen Betriebssystemblöcke

LOG_CHECKPOINT_TIMEOUT
Gibt an, wie oft ein Checkpoint ausgelöst wird, basierend auf Zeit seit dem letzten Checkpoint

FAST_START_IO_TARGET
Gibt die maximale Anzahl von IO-Vorgängen für ein Rollforward an, die im Fehlerfall durchgeführt werden müssten. Ist diese Anzahl überschritten, so werden die notwendigen Schritte (Schreiben von Dirty Blocks auf Platte und Setzen von Checkpoints) durchgeführt, um die Anzahl der IO's für ein Rollforward wieder unter den Wert von FAST_START_IO_TARGET zu bringen.

2.25.9 Block Media Recovery

Wenn nur bestimmte Blöcke einer Datei beschädigt sind, so können Sie ein sogenanntes Block Media Recovery durchführen. Hierbei werden nur die Blöcke, die defekt sind, zurückgespielt. Die Datendatei kann dabei Online sein. Hier ein Beispiel:

```
blockrecover datafile 12 block 1,2,3,4,5 datafile 32 block 10120,10121
```

2.25.10 Retention Policy

Mit Hilfe der Retention Policy ist es möglich sicherzustellen, dass Sicherungen bis zu einem bestimmten Zeitpunkt in der Vergangenheit verfügbar sind. Mit dem folgenden Befehl setzen Sie die Retention Policy auf 28 Tage. Alle älteren Sicherungen werden als Obsolete gekennzeichnet.

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 28 DAYS;
```

Nun können Sie alle älteren Sicherungen löschen. Hierfür müssen Sie zuvor einen Kanal für die Wartung zuweisen und im Anschluss wieder freigeben:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
DELETE OBSOLETE;
RELEASE CHANNEL;
```

Hinweis: Die Default-Retention-Policy entspricht der REDUNDANCY von 1. Das bedeutet, dass 1 Sicherungs-Set jeweils aufbewahrt wird. Durch folgenden Befehl wird die Retention Policy wieder auf Ihren ursprünglichen Default-Wert (REDUNDANCY 1) gesetzt.

```
CONFIGURE RETENTION POLICY CLEAR;
```

2.25.11 Übung Vollsicherung mit RMAN

Übungen siehe Seite 278.

Kapitel 3

High-Performance-Tuning

3.1 Tools und der Optimierer

3.1.1 Anzeigen des Ausführungsplanes

Vorteil: Leichtes Erkennen des Ausführungsplanes
Nachteil: Keine genaueren Angaben (wie bei tkprof) Erstellen einer Plan-Tabelle

```
start /oracle/ora92/rdbms/admin/utlxplan.sql
```

Einschalten

```
set autotrace on explain;
```

Ausschalten

```
set autotrace off;
```

3.1.2 TKPROF

Das Aktivieren des Tracings erfolgt durch Anpassen des Parameters SQL_TRACE in der init-Datei oder mit

```
alter system set sql_trace = true;
```

bzw.

```
alter session set sql_trace = true;
```

Der Speicherort des Trace-File wird durch den Parameter user_dump_dest definiert. Die aktuellen Parameter lassen sich anzeigen.

```
show parameters;
```

Das Trace-File ist schwer lesbar. Es kann ein eine lesbare Datei umgewandelt werden.

```
tkprof /oracle/admin/pingudb/udump/pingudb_ora_2404.trc t1.txt
```

Anmerkung: TIMED_STATISTICS sollte auf TRUE gestellt sein.

3.1.3 Der Cursor

Speicherbereich mit

- SQL-Anweisung: PARSED, UNPARSED
- Ausführungsplan
- Pointer auf die aktuelle Zeile

3.1.4 Das Parsing

Folgendes wird ausgeführt:

- Syntax-Check
- Regel-Check (z. B. Keine Gruppenfunktion etc)
- Semantik-Check: Existenz der Spalten etc.
- Security-Check
- Ermitteln des Ausführungsplanes

3.1.5 Binden der Variablen

Was passiert während des Bindings?

```
VARIABLE emp_num NUMBER
BEGIN
 emp_num:=1;
END;

SELECT firstname FROM employees
  WHERE employeeid=:emp_num
;
```

3.1.6 EXECUTE und FETCH

Was passiert während des Executing / Fetch?

- DML = Execute
- SELECT = FETCH
  - Array-Fetch = mehr als eine Zeile
  - Result-Set = Ergebnismenge

3.1.7 Übung – Tools zum Performance-Tuning

Übungen siehe Seite 279.

3.2 Verwalten von Indizes

Indexarten und deren Nutzen siehe Seite 127.

3.2.1 Rebuilding von Indizes

Das Rebuilding von Indizes wird verwendet, um

- einen Index auf einen anderen Tablespace zu verschieben,
- gelöschte Einträge zu 'neutralisieren',
- einen reverse key Index in einen B-tree Index und vice versa umzuwandeln.

```
ALTER INDEX scott.ord_region_id_idx REBUILD TABLESPACE indx02;
```

3.2.2 Recreation versus Rebuilding

Drop und Recreate

- Ein Umbenennen ist möglich.
- Eine Umwandlung von UNIQUE in NONUNIQUE und vice versa ist möglich.
- Eine Umwandlung von B-Tree in Bitmap und vice versa ist möglich.
- Es wird kein weiterer Speicherplatz benötigt.
- Eine Sortierung ist erforderlich.
- Der Index ist zeitweise nicht verfügbar.
- Falls der Index als Unterstützung für einen Constraint benötigt wird, ist Drop und Recreate nicht möglich.

Rebuild

- Ein Umbenennen ist nicht möglich.
- Umwandlung von UNIQUE in NONUNIQUE und vice versa ist nicht möglich.
- Eine Umwandlung von B-Tree in Bitmap und vice versa ist nicht möglich.
- Es wird Speicherplatz für ein Duplikat in einem temporären Segment gebraucht.
- Eine Sortierung ist nicht erforderlich.
- Der Index ist immer verfügbar.
- Rebuild ist immer möglich.

3.2.3 Überprüfen der Index-Gültigkeit

Tabelle INDEX_STATS wird gefüllt:

```
ANALYZE INDEX scott.ord_region_id_idx VALIDATE STRUCTURE;
```

LF_ROWS_LEN = Länge aller Datensätze (LEAF)
DEL_LF_ROWS_LEN = Länge aller gelöschte Datensätze (LEAF)

```
select (del_lf_rows_len/lf_rows_len)*100 from index_stats;
```

Ist das Ergebnis größer als 20%. sollte ein Rebuild des Index erfolgen.

3.2.4 Übung Index-Verwaltung

Übungen siehe Seite 283.

3.3 Optimierungen

3.3.1 Access-Methoden

Table-Scan	Eher schlecht, außer bei kleinen Tabellen
Rowid-Access	File-Nr + Block-Nr + Row-Nr (AAAB0?..)
Index-Lookup	Bevorzugte Methode bei großen selektiven Tabellen
Hash Key Access	Schnelle Zugriffsmethode bei großen Tabellen

3.3.2 Join-Methoden

Sort-Merge

- Zwei sortierte Haufen

Nested Loops

- Full Table Scan einer Tabelle
- Index-Scan der anderen Tabelle

Hash Join

- Für die größere Tabelle wird eine Art 'On The Fly Index' erstellt

3.3.3 Der Optimierer

Regelbasiert	eher veraltet
Kostenbasiert	aktuell und möglichst zu bevorzugen

3.3.4 Der Optimierungsmodus

CHOOSE	Cost, wenn Tabellen analysiert sind. Rule, wenn Tabellen nicht analysiert sind.
RULE	Es wird immer Rule verwendet.
ALL_ROWS	Cost, optimiert für die Rückgabe aller Datensätze der Ergebnismenge.
FIRST_ROWS	Cost, optimier für die Rückgabe des ersten Datensatzes der Ergebnismenge.

Standard ist CHOOSE. Feststellen läßt sich die Einstellung mit

```
show parameters;
```

Ändern in der init.ora oder mit

```
ALTER SESSION SET OPTMIZER_GOAL=RULE;
```

3.3.5 Hints – Erzwingen einer Methode

```
SELECT /*+RULE*/ employeeid FROM employees;
```

```
SELECT /*+ALL_ROWS*/ employeeid FROM employees;
```

3.3.6 Analyse von Tabellen / Indizes

```
analyze table employees compute statistics;

analyze table employees estimate statistics sample 10 percent,

analyze table employees delete statistics;
```

3.3.7 Experiment 1

Voraussetzung:
Existenz der Tabelle order_details und `set autotrace on exp;`
Quantity = 10 bei 50% der Datensätze
Quantity = 20 bei 50% der Datensätze

Auf Quantity steht ein Index –>Wenig selektiv.

```
select /*+RULE*/ * from order_details where quantity=10;
select /*+ALL_ROWS*/ * from order_details where quantity=10;

analyze table order_details compute statistics;

select /*+RULE*/ * from order_details where quantity=10;

select/*+ALL_ROWS*/ * from order_details where quantity=10;
```

3.3.8 Experiment 2

Es wird die Tabelle auf mehr als 1 Mio. Datensätze vergrößert.

```
--Es wird ein Datensatz mit Quantity=30 hinzugefügt
insert into order_details values(11000,10,10,30,0);

select /*+RULE*/ * from order_details where quantity=10;
select /*+RULE*/ * from order_details where quantity=30;

select /*+ALL_ROWS*/ * from order_details where quantity=10;
select /*+ ALL_ROWS */ * from order_details where quantity=30;
```

3.3.9 Experiment 3

Es werden 10 Buckets gebildet (hier würden 3 ausreichen, da nur drei Werte in quantity stehen).

```
--Erstellen eines Histogramms
analyze table order_details
  compute statistics for columns quantity
  size 10
;

select /*+RULE*/ * from order_details where quantity=10;
select /*+RULE*/ * from order_details where quantity=30;

select /*+ALL_ROWS*/ * from order_details where quantity=10;
select /*+ ALL_ROWS */ * from order_details where quantity=30;
```

3.3.10 Übung Optimierung

Übungen siehe Seite 287.

3.4 Stored Outlines

3.4.1 Gespeicherte Ausführungspläne

```
create or replace outline o1 for category cat1 on
  select * from order_details where quantity=:quant
;
```

3.4.2 Aktivieren der erstellten Stored Outline

```
alter session set use_stored_outlines=cat1;
```

3.4.3 Das Experiment

Erstellen einer Stored Outline, die einen Index-Scan durchführen wird

```
analyze table order_details delete statistics;

select orderid from order_details where quantity=10;
```

Index-Scan, da Rule-Optimizer (es gibt keine Analyse).

```
create outline o1 for category cat1 on
  select orderid from order_details where quantity=10;

alter session set use_stored_outlines = cat1;

analyze table order_details compute statistics;

select orderid from order_details where quantity=10;
```

Immer noch Index-Scan, obgleich der Cost-Optimizer Full-Scan bevorzugt hätte.

```
select orderid from order_details where quantity=20;
```

Full-Scan, da kein exaktes Matching.

```
alter session set use_stored_outlines = false;

select orderid from order_details where quantity=10;
```

Cost-Optimizer entscheidet sich für Full-Scan.

3.4.4 Übung Stored Outlines

Übungen siehe Seite 291.

3.4.5 Übung Otimierungen

Übungen siehe Seite 293.

3.5 Materialized Views

Materialized Views (Materialisierte Sichten) sind Tabellen mit Daten basierend auf einer Abfrage.

- Beschleunigt den Zugriff
- Verbreitet in Datawarehouses
- Verbreitet als Snapshot

3.5.1 Beispiel

Gegeben: 1 Million order_details
Hinweis: ANALYSE nicht vergessen
Folgende Abfrage dauert ca. 15 Sekunden:

```
select orderid,count(*)
  from order_details
  group by orderid
;
```

Nach Erstellen einer Materialized Views ist die Abfragedauer lediglich 0,07 Sekunden.

3.5.2 CREATE MATERIALIZED VIEW

Vor dem Erstellen muß die entsprechende Tabelle analysiert werden.

```
analyze table order_details compute statistics;
```

```
create materialized view mv1
  build immediate
  refresh on commit
  enable query rewrite
  as
    select orderid,count(*) from
      order_details group by orderid
;
```

3.5.3 BUILD

IMMEDIATE	sofort
DEFERRED	nach dem ersten Refresh

3.5.4 REFRESH

COMPLETE	vollständiges Neuerstellen (default)
FAST	nur die Veränderungen Bei FAST muß ein Log erstellt werden.
COMMIT	nach jedem Commit
REFRESH	geschieht durch `dbms_snapshot.refresh('mv1')`

3.5.5 Voraussetzungen

Berechtigung:

```
grant CREATE MATERIALIZED VIEW to user;
```

Analysieren der Tabelle für den Cost-Based Optimizer.

```
analyze table tabelle compute statistics;
```

Parameter

```
alter session set query_rewrite_enabled=true;
```

```
alter session set query_rewrite_enabled=true;
alter session set query_rewrite_integrity=enforced | stale_tolerated;
```

stale_tolerated	auch veraltete Daten möglich
enforced	keine veralteten Daten möglich

Hilfreiche Einstellungen

```
set timing on;
set autotrace on exp;
```

3.5.6 Übung Materialized Views

Übungen siehe Seite 299.

3.6 Einführung in den Resourcen-Plan

Ziel: Effektive Ressourcenplanung
Weg: Gruppenbildung und Zuweisen von Ressourcen zu den Gruppen

3.6.1 Beispiel

- Gruppe OLTP_USERS
  Benötigen viel CPU-Ressourcen
- Gruppe BATCH_USERS
  Benötigen wenig CPU-Ressourcen
  (um die OLTP-User nicht zu stören)
- Gruppe ADHOC_USERS
  Benötigen wenig Ressourcen
- Gruppe OTHER_GROUPS
  Alle anderen (benötigt fast keine Ressourcen)
  Ist standardmäßig vorhanden

CPU-Nutzung bei RM

- LEVEL 1
  OLTP erhält 80% der CPU
- LEVEL 2
  ADHOC_USERS erhalten von den restlichen 20% insgesamt 80%, also 16%.
- LEVEL 2
  BATCH_USERS erhalten von den restlichen 20% insgesamt 10%, also 2%.
- LEVEL 3
  OTHER_USERS erhalten von den restlichen 2% insgesamt 100%, also 2%.

Anmerkung:
Gilt nur, wenn die jeweils vorigen Level Ihre Grenze ausgeschöpft haben. Wenn OLTP im Augenblick gar nichts machen, geht natürlich 80% der gesamten CPU an die ADHOC_USER.

Ressourcen Gruppen	**Level 1 CPU**	**Level 2 CPU**	**Level 3 CPU**
OLTP_USERS	80%	0%	0%
BATCH_USERS	0%	10%	0%
ADHOC_USERS	0%	80%	0%
OTHER_GROUPS	0%	0%	100%

3.6.2 Praktische Umsetzung

- Durch Anweisungen.
- Mit dem OEM-Console (Vorsicht: Buggy).

3.6.3 dbms_resource_manager

Erstellen

```
exec dbms_resource_manager.clear_pending_area();
exec dbms_resource_manager.create_pending_area();


exec dbms_resource_manager.create_plan
 (plan => 'Plan_1',comment=>'Mein erster Plan')
;


exec dbms_resource_manager.create_consumer_group
  (consumer_group => 'OLTP_USERS',comment=>'Meine erste Gruppe')
;
exec dbms_resource_manager.create_consumer_group
  (consumer_group => 'ADHOC_USERS',comment=>'')
;
exec dbms_resource_manager.create_consumer_group
  (consumer_group => 'BATCH_USERS',comment=>'')
;


exec dbms_resource_manager.create_plan_directive
  (plan => 'Plan_1', group_or_subplan => 'OLTP_USERS', cpu_p1 => 80,comment=>'')
;
exec dbms_resource_manager.create_plan_directive
  (plan => 'Plan_1', group_or_subplan => 'ADHOC_USERS', cpu_p2 => 80,comment=>'')
;
exec dbms_resource_manager.create_plan_directive
  (plan => 'Plan_1', group_or_subplan => 'BATCH_USERS', cpu_p2 => 10,comment=>'')
;
exec dbms_resource_manager.create_plan_directive
  (plan => 'Plan_1', group_or_subplan => 'OTHER_GROUPS', cpu_p3 => 100,comment=>'')
;


exec dbms_resource_manager.submit_pending_area();


exec dbms_resource_manager.create_pending_area();
exec dbms_resource_manager_privs.grant_switch_consumer_group
  (grantee_name => 'ADHOC_1',consumer_group => 'ADHOC_USERS', grant_option => FALSE)
;


exec dbms_resource_manager.set_initial_consumer_group
  (user => 'ADHOC_1',consumer_group => 'ADHOC_USERS')
;



alter system set resource_manager_plan = 'Plan_1';
```

Hinweis:
grant switch consumer group ist Voraussetzung für set initial consumer group.
OTHER_GROUPS muß es geben, da dort alle, die in keiner anderen Gruppe sind, stehen.

Änderungen

```
exec dbms_resource_manager.clear_pending_area();
exec dbms_resource_manager.create_pending_area();


exec dbms_resource_manager.update_plan_directive
  (plan => 'Plan_1', group_or_subplan => 'OLTP_USERS', new_cpu_p1 => 70,new_comment=>'')
;
exec dbms_resource_manager.submit_pending_area();
```

Hinweise:
alle Parameter heißen jetzt new_
Unlimited ist -1
Wichtige Parameter können mit dem OEM ermittelt werden.

3.6.4 Aktivieren / Deaktiveren

Es kann nur ein Plan aktiv sein.

```
alter system set resource_manager_plan = 'Plan_1';


alter system set resource_manager_plan = '';
```

In der INIT-Datei durch resource_manager_plan = 'Plan_1'

3.6.5 Weitere Einstellungen

Abfragenlänge: Wenn geschätzte Ausführungszeit größer ist, dann erfolgt Abbruch.
TABLE muß analysiert worden sein.

Maximale Sessionzahl je Gruppe
Maximale Undo-Größe

3.6.6 Übung Resourcen-Plan

Übungen siehe Seite 303.

3.7 Tuning des Database Buffer Caches

3.7.1 Database Buffer Cache

Wichtige Parameter sind ON THE FLY änderbar.

```
alter system set DB_CACHE_SIZE = 64M;
alter system set DB_KEEP_CACHE_SIZE = 100K;
alter system set SHARED_POOL_SIZE = 48M;
```

Weitere Parameter erfordern Neustart.

```
alter system set SGA_MAX_SIZE=150M scope=spfile;
```

Informatione über Größe stehen in V$BUFFER_POOL.

Buffer Cache:
<128MB: 4MB Granular
>128MB: 16MB Granular

Verwaltung durch zwei Listen

- Write-List
- LRU-List

3.7.2 Latches

Latches sind Sperren auf niederer Ebene.

- Werden für den Zugriff auf die Pools vergeben (bis zu insgesamt DB_BLOCK_LRU_LATCHES).
- Ein Latch schützt mindestens 50 Blöcke.
- Sollten ab 9i nicht mehr per Hand verändert werden.

3.7.3 Aufbau der SGA

System Global Area

- Buffer Cache
  - Keep
  - Recycled
  - Default
- Shared Pool
  - Library Cache
- Java Pool
- Large Pool
- Redo-Log-Buffer

Die Summe kann durch folgende Formel ermittelt werden:

```
DB_CACHE_SIZE
+ DB_KEEP_CACHE_SIZE
+ DB_RECYCLE_CACHE_SIZE
+ DB_nk_CACHE_SIZE
+ SHARED_POOL_SIZE
+ LARGE_POOL_SIZE
+ JAVA_POOL_SIZE
+ LOG_BUFFERS
+ 1MB

select 1, name, to_number(value/1024/1024) value
  from v$parameter
    where upper(name)
      like 'DB%CACHE_SIZE' or upper(name) in
        ('SHARED_POOL_SIZE','LARGE_POOL_SIZE','JAVA_POOL_SIZE','LOG_BUFFER')
      union
        select 1, '+ 1MB', 1 from dual order by 2;
```

Beispiel

```
NAME     VALUE
----------------------------
+ 1MB                     1
db_cache_size            32
db_keep_cache_size        0
db_recycle_cache_size     0
db_16k_cache_size         0
db_2k_cache_size          0
db_32k_cache_size         0
db_4k_cache_size          0
db_8k_cache_size          0
java_pool_size           32
large_pool_size           0
log_buffer                ,5
shared_pool_size         88
```

3.7.4 Tuning des Database Buffer Caches

Messen von Cache Hit Ratio je Instanz

Physical Reads, DB Block Gets und Consistent Gets stehen in V$SYSSTAT seit Instanzstart.
Cache Miss Ratio = physical reads/(db block gets + consistent gets)
Cache Hit Ratio = 1-physical reads/(db block gets + consistent gets)
Formel:

```
select 1-value/
      (
        select sum(value)
          from v$sysstat
          where lower(name) in ('consistent gets','db block gets')
       )
  from v$sysstat
  where lower(name) in ('physical reads')
;
```

Physical Reads sind von Disk gelesene Datenblöcke.
DB Block Gets sind im Cache gefundene Datenblöcke.
Consistent Gets sind im Cache gefundene Datenblöcke für die Lesekonsistenz.

Empfehlung:
Sollte über 0,9 liegen (hängt aber von den Umständen ab), z.B. großer Buffer-Cache – mehrmals select * from scott.emp – Hit-Ratio steigt.

Messen von Cache Hit Ratio je Pool

```
select name,
    100-round((physical_reads/(db_block_gets+consistent_gets))*100,2)
    as Hit_Ratio
  from v$buffer_pool_statistics
;
```

Achtung: Alle Buffer (keep, recycle und default), die definiert wurden, müssen auch ungleich 0 sein, ansonsten ergibt es eine Division by Zero. Laden einer Tabelle in den entsprechenden Pool (alter table scott.dept storage buffer_pool KEEP) und Abfragen dieser Tabelle (select * from scott.dept).

Messen von Cache Hit Ratio je Session

```
select username,
       osuser,
       1-(io.physical_reads/(io.block_gets+io.consistent_gets)) "Hit Ratio"
  from v$sess_io io, v$session sess
  where io.sid = sess.sid
    and (io.block_gets + io.consistent_gets)!=0
    and username is not null
;
```

Beispiel Die Datenbankperformance ist gut, nur ein Benutzer klagt.
Empfehlung: Sollte bei jedem Benutzer über 0,9 liegen

3.7.5 Lösungsansätze

Nutzen von verschiedenen Pool-Typen

- Keep Pool
  Informationen, die im Database Buffer Cache lange bleiben sollten.
- Recycle Pool
  Informationen, die im Database Buffer Cache nicht lange bleiben sollten.
- Default Pool
  Informationen, die zu keinem der anderen beiden passen.

```
alter table tablename storage buffer_pool keep | recycle;
```

3.7.6 Definition der Pool-Typen

Parameter der entsprechenden INIT-Datei:

```
DB_BLOCK_BUFFERS =              1000
BUFFER_POOL_KEEP =               400
BUFFER_POOL_RECYCLE =            100
DEFAULT POOL = 1000-(400+100) = 500
```

Speichern einer Tabelle in einem speziellen Pool:

```
ALTER TABLE test STORAGE (buffer_pool KEEP);
```

3.7.7 Cache-Advice

Gibt Empfehlungen für den Buffer-Cache.

Anschalten

```
alter system set db_cache_advice=on;
```

Eine Weile laufen lassen

Abfragen

```
select distinct size_for_estimate,estd_physical_reads
  from v$db_cache_advice;
```

Ausschalten

```
alter system set db_cache_advice=off;
```

3.7.8 Cache-Option und Hint

Cache

Blöcke werden an der MRU-Seite gecacht. Dies ist sinnvoll bei häufigen Full-Table-Scan der gleichen Tabelle.

Cache-Hint

```
SELECT /*+ CACHE(scott.emp) */ empno, ename FROM scott.emp;
```

CREATE TABLE CACHE

```
create table off_car(carnr int,carname varchar2(20)) CACHE;
```

3.7.9 Guidelines

Gute Kandidaten für KEEP-Pool:
Kleine, häufig frequentierte Objekte auf die mit Full Table Scan FTS zugegriffen wird.

Gute Kandidaten für RECYCLE-Pool:
Große, selten frequentierte Objekte.

Gute Kandidaten für DEFAULT-Pool:
Alle anderen Objekte.

3.7.10 Übung Datenbank-Tuning

Übungen siehe Seite 306.

3.8 Undo / Rollback Segments (Before Images)

3.8.1 Zweck

Read Consistency
Transaction Rollback
Transaction Recovery

3.8.2 Planung der Anzahl der Rollback Segments

OLTP

Kleine, aber viele Rollbacksegmente
Ein RBS für je 4 Transaktionen

Batch

Groß, aber wenig

3.8.3 Undo Management

Voraussetzung: Undo-Tablespace(s).

```
CREATE UNDO TABLESPACE undots1
  DATAFILE '/oracle/oradata/ORCL/orcl_undots101.dbf'
  SIZE 100M
  AUTOEXTEND ON
;
```

```
UNDO_MANAGEMENT (MANUAL or AUTO)
```

Automatic

```
alter system set undo_management=auto | manual scope=spfile;
```

```
UNDO_RETENTION (time in seconds) AUM
```

Manual

```
alter system set undo_management=manual scope=spfile;
```

3.8.4 Die Erstellung von Rollback Segmenten

```
CREATE ROLLBACK SEGMENT rbs01 TABLESPACE undotbs1;
```

Achtung:

1. Angabe von Minextents / Maxextents etc. nur bei DMT möglich
2. DMT nur möglich, wenn System-TS auch DMT ist

3.8.5 Online-Setzen

```
ALTER ROLLBACK SEGMENT rbs01 ONLINE;
```

3.8.6 Deallocating Space

```
ALTER ROLLBACK SEGMENT rbs01 SHRINK TO 4M;
```

3.8.7 Zuweisen eines bestimmten Rollback-Segmentes

```
set transaction use rollback segment rbs01;
```

3.8.8 Offline-Setzen

Rollback Segmente werden Offline gesetzt, um sie löschen zu können.

```
ALTER ROLLBACK SEGMENT rbs01 OFFLINE;
```

3.8.9 Löschen von Rollback Segmenten

```
DROP ROLLBACK SEGMENT rbs01;
```

3.8.10 Wichtige Sichten

V$ROLLSTAT

Prozentsatz von 'Warten auf ein Rollbacksegment'

```
select round(sum(waits)/sum(gets),2) from v$rollstat;
```

Wenn >1 %, dann mehr Rollback-Segmente.

V$WAITSTAT

Prozentsatz von 'Warten auf Zugriff auf einen bestimmten Rollback-Segment-Block'.

```
select class, count
  from v$waitstat
  where class
   in
   ('system undo header', 'system undo block', 'undo header', 'undo block')
;
```

Logische Lesevorgänge

```
select sum(value)
  from v$sysstat
  where name
    in ('db block gets', 'consistent gets')
;
```

Wenn count größer als 1% für irgend eine Class, dann mehr Rollback-Segmente.

3.8.11 Übung Undo / Rollback Segments

Übungen siehe Seite 312.

3.9 StatsPack

StatsPack löst utlbstat und utlestat ab.

3.9.1 Installation

```
start /oracle/ora92/rdbms/admin/spcreate.sql
...
Geben Sie einen Wert für perfstat_password ein: pinguin
...
Geben Sie einen Wert für default_tablespace ein: TOOLS
...
Geben Sie einen Wert für temporary_tablespace ein: TEMP
```

Bei Fehlern ggf. die Datei spcreate.sql anpassen:

```
-- connect perfstat/&&perfstat_password
connect perfstat/pinguin@testdb29
```

Bei Fehlern erst alles wieder löschen bevor wieder spcreate.sql aufgerufen wird.

```
start /oracle/ora92/rdbms/admin/spdrop.sql
```

3.9.2 Snapshot erzeugen

```
connect perfstat/pinguin@testdb29
execute statspack.snap
```

3.9.3 Snapshots abfragen

```
select snap_id, snap_time from STATS$SNAPSHOT;
```

3.9.4 Bericht erzeugen

```
start /oracle/ora92/rdbms/admin/spreport.sql
```

3.9.5 Übung StatsPack

Übungen siehe Seite 312.

3.10 Cluster und Index-Organized Tables

3.10.1 Verteilung von Zeilen in einer Tabelle

```
Table      Cluster     Index-organized Table

---------- Ordering of Rows --------->

Random     Grouped     Ordered
```

3.10.2 Clusters

Unclustered ORD und ITEM Tabellen

```
ORD_NO PROD QTY ...
-----------------
101 A4102 20
102 A2091 11
102 G7830  20
102 N9587  26
101 A5675 19
101 W0824 10


ORD_NO ORD_DT   CUST_CD
------------   ------
101 05-JAN-97    R01
102 07-JAN-97    N45
```

Clustered ORD und ITEM Tabellen

```
Cluster Key
(ORD_NO)
101    ORD_DT       CUST_CD
        05-JAN-97    R01
             PROD    QTY
             A4102   20
             A5675   19
             W0824   10
102    ORD_DT       CUST_CD
       07-JAN-97     N45
             PROD    QTY
             A2091   11
             G7830   20
             N9587   26
```

Cluster Arten

Index Cluster

Hash Cluster

Erstellen eines Index Clusters

1. Cluster erstellen.

```
CREATE CLUSTER scott.ord_clu (ord_no NUMBER(3))
   SIZE 200 TABLESPACE DATA01
   STORAGE(INITIAL 5M NEXT 5M PCTINCREASE 0)
;
```

2. Cluster-Index erstellen.

```
CREATE INDEX scott.ord_clu_idx
  ON CLUSTER scott.ord_clu
  TABLESPACE INDX01
  STORAGE(INITIAL 1M NEXT 1M PCTINCREASE 0)
;
```

3. Tabellen im Cluster erstellen.

```
CREATE TABLE scott.ord
  (
   ord_no NUMBER(3)
   CONSTRAINT ord_pk PRIMARY KEY,
   ord_dt DATE, cust_cd VARCHAR2(3)
   )
  CLUSTER scott.ord_clu(ord_no)
;
CREATE TABLE scott.item
  (
    ord_no NUMBER(3)
    CONSTRAINT item_ord_fk REFERENCES scott.ord,
    prod VARCHAR2(5),
    qty NUMBER(3),
    CONSTRAINT item_pk PRIMARY KEY(ord_no,prod)
   )
  CLUSTER scott.ord_clu(ord_no)
;
```

Erstellen eines Hash Clusters

1. Cluster erstellen.

```
CREATE CLUSTER scott.off_clu
  (
   country VARCHAR2(2),postcode VARCHAR2(8)
   )
  SIZE 500 HASHKEYS 1000
  TABLESPACE DATA01
  STORAGE(INITIAL 5M NEXT 5M PCTINCREASE 0);
```

2. Tabellen im Cluster erstellen.

```
CREATE TABLE scott.office
  (
   office_cd NUMBER(3),
   cost_ctr NUMBER(3),
   country VARCHAR2(2),
   postcode VARCHAR2(8)
   )
  CLUSTER scott.off_clu(country,postcode)
;
```

Dropping Clusters

Nutze INCLUDING TABLES für das Löschen der Tabellen und des Clusters.

```
DROP CLUSTER scott.ord_clu INCLUDING TABLES;
```

Oder lösche die Tabellen vor dem Cluster.

```
DROP TABLE scott.ord;
DROP TABLE scott.item;
DROP CLUSTER scott.ord_clu;
```

3.10.3 Index-Organized Tables

Indexed access on table:
ROWID

Accessing index-organized table:
Non-key columns
Key column
Row header

Creating Index-Organized Tables

```
CREATE TABLE scott.sales
  (
   office_cd  NUMBER(3),
   qtr_end    DATE,
   revenue    NUMBER(10,2),
   review     VARCHAR2(1000),
   CONSTRAINT sales_pk
   PRIMARY KEY(office_code, qtr_end)
  )
  ORGANIZATION INDEX TABLESPACE data01
  PCTTHRESHOLD 20
  OVERFLOW TABLESPACE data02
;
```

Row Overflow IOT tablespace
Overflow tablespace

3.10.4 Übung Index-Cluster

Übungen siehe Seite 316.

Kapitel 4

Backend / Frontend

4.1 PHP

Apache starten.

```
apachectl start
```

Testen, ob PHP in Apache eingebunden ist.
Im Dokumentenverzeichnis des Servers die Datei info.php mit einem Editor anlegen und folgende Zeilen eintragen:

```
<?
phpinfo();
?>
```

Im Browser aufrufen:

```
http://127.0.0.1/info.php
```

Erscheinen umfangreiche Informationen zur Konfiguration von PHP, dann ist alles OK.

4.1.1 PHP – Oracle

Ein einfaches PHP-Beispiel zur Datenbankanbindung zu Oracle.

```
<html>
<body>

<?php
// open a connection
if (!$db = @ora_logon("scott@testdb29","tiger")) {
   $error = ora_error();
   printf("There was an connecting error. Error was: %s", $error);
   die();
}
else  echo "Connect...\n<br>";
$curs = ora_open($db);
$sql = "SELECT * FROM dept";

// check is that SQL statement is.
if (!@ora_parse($curs,$sql)) {
   echo "Error in parse. Error was :", ora_error($curs);
} else {
   ora_exec($curs);

   // display results using the column offset
   while (ora_fetch_into($curs, $results)) {
      echo $results[0];
      ?>...<?php
      echo $results[1];
      ?><br><?php
   }
}
?>

</body>
</html>
```

4.1.2 PHP – mySQL

LAMP
PHP wird als Modul in den Apache-Webserver eingebunden. Meist wird als Datenbank dazu mySQL verwendet. Die Kombination Linux + Apache + mySQL + PHP wird als LAMP bezeichnet.

mySQL starten (SuSE-Linux):

```
rcmysql start
```

Ein einfaches PHP-Beispiel zur Datenbankanbindung zu mySQL.

```
<html>
<head><title>php</title></head>
<body>
<?php
////////////////////////////////////////////////////////////////////////////
//Verbindung zu mysql
//--------------------
//Verbindung mit MySQL:
//resource mysql_connect([string Server[,string User[,string Passwort]]])
//
//Auswahl der Datenbank:
//bool mysql_select_db(string Datenbankname[,resource Verbindungs-Kennung])
//
//Eine SQL Abfrage abschicken:
//resource mysql_query(string Anfrage[,resource Verbindungs-Kennung])
//
//Das Ergebnis der Abfrage auswerten:
//array mysql_fetch_assoc(resource Ergebnis)
//Gibt eine Zeile als assoziatives Array aus. Geht dann zur nächsten Zeile.
//
//array mysql_fetch_row(resource Ergebnis-Kennung)
//Gibt eine Zeile als indiziertes Array aus. Geht dann zur nächsten Zeile.
//
//Fehler ausgeben:
//string mysql_error([resource Verbindungs-Kennung])
////
////////////////////////////////////////////////////////////////////////////
////
$dbh = mysql_connect("localhost", "root", "pinguin")
  or die ("keine Verbindung möglich");

mysql_select_db ( "mysql" , $dbh );

$sql = "select * from user";
$ergebnis = mysql_query ( $sql , $dbh) or die (mysql_error());

while ($zeile = mysql_fetch_assoc ( $ergebnis)) {
        print_r($zeile);
        echo "<br>";
}
//
////////////////////////////////////////////////////////////////////////////
php?>
</body>
</html>
```

4.2 Java

4.2.1 Eine kleine DB in Java

Dieses Programms soll Teilnehmer einer Schulungsfirma verwalten können. Schreiben Sie ein Programm, welches aus folgenden Klassen bestehen soll:

Klasse Teilnehmer
Attribute (alle private, Typ String): Name, Vorname, Kurs
Methoden: getName(...), setName(), getVorname(), ...

Klasse Datenausgabe
Methode: printDaten() ... mit System.out.println()
Über diese Methode sollen alle Teilnehmer-Objekte ausgegeben werden.

Klasse T_Verwaltung
Methode: main(...)

Aufgabe:
Kreieren Sie alle Klassen.
Legen Sie in dieser Klasse T_Verwaltung zunächst zwei Objekte der Klasse Teilnehmer an z.B. 'Meier, Kurt' Netzwerk-Administrator und 'Bayer, Astrid' DB-Administrator.
Geben Sie alle Daten aus.
Verändern Sie den Kurs des Teilnehmers Meier.
Geben Sie alle Daten aus.

Eine Lösung:

```
class Teilnehmer {
     private String name, vorname, kurs;
     public Teilnehmer(String n, String v, String k) {
        this.name = n;
        this.vorname = v;
        this.kurs = k;
     }
      public String getName() {
        return name;
     }
     public void setName(String n) {
        name=n;
     }
     public String getVorName() {
        return vorname;
     }
     public void setVorName(String v) {
        vorname=v;
     }
     public String getKurs() {
        return kurs;
     }
     public void setKurs(String k) {
        kurs=k;
     }
}
class Datenausgabe {
     // Über diese Methode sollen alle Teilnehmer-Objekte ausgegeben werden.
     public static void printDaten(Teilnehmer te) {
        System.out.println("Name: " + te.getName());
        System.out.println("Vorname: " + te.getVorName());
        System.out.println("Kurs: " + te.getKurs());
        System.out.println();
     }
}
class T_Verwaltung {
       public static void main(String[] args) {
          System.out.println("Verwaltung der Teilnehmer einer Schulungsfirma");
          Teilnehmer t1 = new Teilnehmer("Meier","Kurt","Netzwerk-Administrator");
          Teilnehmer t2 = new Teilnehmer("Bayer","Astrid","DB-Administrator");
          Datenausgabe.printDaten(t1);
          Datenausgabe.printDaten(t2);
          System.out.println("");
          System.out.println("Meiser wechselt zum Kurs DB-Administrator");
          System.out.println("");
          t1.setKurs("DB-Administrator");
          Datenausgabe.printDaten(t1);
          Datenausgabe.printDaten(t2);
       }
}
```

4.2.2 JDBC

JDBC-Treibertypen

Typ 1 JDBC-ODBC-Brücken

Typ 2 Treiber, die nur teilweise in Java programmiert sind. Übersetzen JDBC-Aufrufe an ein Client-Api für Oracle, Sybase, Informix, DB2 ...

Typ 3 Treiber, die einen Java-Client verwenden und über das Netz mit einen Middleware-Server kommunizieren, der ein DB-unabhängiges Protokoll verwendet. Dieser reicht die Client-Anfrage an den DB-Server weiter.
3-Schichten-Modell (flexibel).

Typ 4 Komplett in Java programmierte Treiber, die ein datenbankspezifisches Netzwerkprotokoll für eine bestimmte Datenquelle implementieren, um darüber direkt über einen offenen IP-Port mit der Datenquelle zu kommunizieren, wie z.B. bei Oracles*Net bzw. Net8.
2-Schichten-Modell (nicht für Access).

Thin-Treiber – Oracle

Lade Thin-Treiber von [10]
Anbindung an Oracle, Lade Oracle-JDBC-Type-4-Treiber von [9]
Oracle8i JDBC Drivers for use with JDK 1.2.x,
JDBC-Thin, 100% Java (classes12.zip, 1.5MB).
Die Datei classes12.zip nicht entzippen, sondern in den CLASSPATH einbinden.

```
set CLASSPATH=$CLASSPATH:.../classes12.zip
```

Eventuell reicht ein Kopieren des JDBC-Treibers nach $JAVA_HOME/jre/lib/ext.

Beispiel:

```
import java.sql.*;

public class Sql_ORA_thin
{
  public static void main( String args[] )
  {
    try {
      Class.forName( "oracle.jdbc.driver.OracleDriver" );
    }
    catch ( Exception e ) {
      System.out.println( "Fehler bei ODBC-JDBC-Bridge" + e );
      return;
    }
    Connection conn;
    Statement stmt;
    ResultSet rSet;

    try {
      // //// Anpassen
      String url = "jdbc:oracle:thin:@192.168.50.29:1521:testdb29";
      conn = DriverManager.getConnection( url, "Scott", "tiger" );
      stmt = conn.createStatement();
      String sqlQuery = "SELECT ename, empno FROM emp";
      rSet = stmt.executeQuery( sqlQuery );
      // ////
    }
    catch ( Exception e ) {
      System.out.println( "Fehler bei Datenbankzugriff" + e );
      return;
    }
    try {
    System.out.println ( "EMANE" + "\t" + "EMPNO");
    System.out.println ( "----------------------");
      while ( rSet.next() )
        System.out.println ( rSet.getString(1) + "\t" + rSet.getString(2));

      stmt.close();
      conn.close();
    }
    catch ( Exception e ) {
      System.out.println( "Fehler bei Tabellenabfrage" + e );
      return;
    }
  }
}
```

ODBC-Treiber – Oracle

Beispiel unter Windows: Eventuell tnsnames.ora anpassen.

ODBC-Treiber laden
Systemsteuerung – Verwaltung – Datenquellen (ODBC)
System-DSN – Hinzufügen: MS ODBC-Treiber für Oracle
Datenquellenname: testdb29 (kann beliebig gewählt werden)
Beschreibung: bla
Benutzername: scott
Server: testdb29 (wie in tnsnames.ora)
Beispiel:

```
import java.sql.*;

public class Sql
{
  public static void main( String args[] )
  {
    try {
      Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
    }
    catch ( Exception e ) {
      System.out.println( "Fehler bei ODBC-JDBC-Bridge" + e );
      return;
    }
    Connection conn;
    Statement stmt;
    ResultSet rSet;
    try {
      // //// Anpassen
      String url = "jdbc:odbc:testdb29";
      conn = DriverManager.getConnection( url, "scott", "tiger" );
      stmt = conn.createStatement();
      String sqlQuery = "SELECT * FROM salgrade";
      rSet = stmt.executeQuery( sqlQuery );
      // ////
    }
    catch ( Exception e ) {
      System.out.println( "Fehler bei Datenbankzugriff" + e );
      return;
    }
    try {
      while ( rSet.next() )
        System.out.println ( rSet.getString(1) + "\t" + rSet.getString(2));
      stmt.close();
      conn.close();
    }
    catch ( Exception e ) {
      System.out.println( "Fehler bei Tabellenabfrage" + e );
      return;
    }
  }
}
```

ODBC-Treiber – mySQL

Beispiel unter Windows
mySQL-ODBC-Treiber von [4] laden.
ODBC-Treiber installieren
Systemsteuerung – Verwaltung - Datenquellen (ODBC)
System-DSN – Hinzufügen mySQL-ODBC-Treiber
Data Source Name: myodbc3-test (kann beliebig gewählt werden)
Description: MySQL ODBC 3.51 TEST DSN
Host/Server: localhost
Database Name: mysql
User: root
Password: pinguin
Port: 3306
SQL-Command: (select * from user)
Test Data Soucre

Beispiel

```
import java.sql.*;

public class MySql
{
  public static void main( String args[] )
  {
    try { Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );}
    catch ( Exception e ) {
      System.out.println( "Fehler bei ODBC-JDBC-Bridge" + e );
      return;
    }
    Connection conn;
    Statement stmt;
    ResultSet rSet;
    try {
      // //// Anpassen
      String url = "jdbc:odbc:meiMySQL";
      conn = DriverManager.getConnection( url, "root", "pinguin" );
      stmt = conn.createStatement();
      String sqlQuery = "SELECT * FROM mysql.user";
      rSet = stmt.executeQuery( sqlQuery );
      // ////
    }
    catch ( Exception e ) {
      System.out.println( "Fehler bei Datenbankzugriff" + e );
      return;
    }
    try {
      while ( rSet.next() )
        System.out.println ( rSet.getString(1) + "\t" + rSet.getString(2));
      stmt.close();
      conn.close();
    }
    catch ( Exception e ) {
      System.out.println( "Fehler bei Tabellenabfrage" + e );
      return;
    }
  }
}
```

4.3 Forms – Kurzeinführung

Developer Suite 10
Forms [32 Bit] Version 9.0.4.0.19 (Production)
Oracle Toolkit Version 9.0.4.0.31 (Production)

4.3.1 Vorbereitung

Server

Eine Oracle-DB muss verfügbar sein. Getestet wurde auf Oracle 8i.
Testdatenbank: Northwind

OC4J (HTTP-Listener) starten

```
$ORA_HOME/j2ee/DevSuite/startinst.sh
```

Client

Formulare werden bei 10 als Applets erstellt, die Verbindung mit einem Servlet aufnehmen: Auf der Clientseite ist nur ein WWW-Browser (Netscape, IE) notwendig. Die Software (JInitiator 1.3.1.13) wird automatisch installiert.

4.3.2 Oracle Forms Builder starten

Anmelden an der Datenbank

Menü Datei - Anmelden
sys/sys@testdb29

Modul erstellen/umbenenen

Beim Starten wird ein Formular 'MODUL1' erstellt. Ist dies nicht vorhanden, kann dies mit Menü Bearbeiten – Erstellen erfolgen. Ein Doppelklick auf 'MODUL1' im Objektnavigator öffnet die Attributpalette. Dort kann der Name 'MODUL1' in 'NORTHWIND_FORMULAR01' geändert werden.

Menü Datei – Speichern unter – NORTHWIND_FORMULAR01.fmb

Block erstellen

Blöcke dienen zur Zuordnung von Formular-Feldern zu Tabellen-Spalten.
Menü Werkzeuge – Datenblock-Assistent – Tabelle oder Ansicht
'Durchsuchen' anklicken. Es erscheinen die Tabellen der Datenbank.
Auswahl 'EMPLOYEES'. Es erscheinen die Spalten der Tabelle.
>>wählt alle Spalten aus.
Tabellenblockname: EMPLOYEES
Datenblock erstellen und danach Layout-Editor aufrufen.

Layout-Assistent

Leinwand: Neue Leinwand (Canvas), Typ: Inhalt
Prompt und Größen können geändert werden.
'Form' auswählen.
Rahmentitel: EMPLOYEES
Es erscheint der Layout-Editor.

Web-Browser festlegen

Menü Bearbeiten – Voreinstellungen – Laufzeit
Web-Browser-Speicherort: Es können nur Mozilla oder MS-IE verwendet werden.

4.3.3 Testen des Formulars

Menü Programm – Form ausführen.
Es wird der Web-Browser getartet.

```
URL: http://localhost:8889/forms90/f90servlet
```

Beim erstten Aufruf wird die Software (JInitiator 1.3.1.13) automatisch installiert.

Anmelden
Es erscheint das Formular. Die kompilierte Modudatei (*.fmx) wird dem Applet als Parameter übergeben.

Datensatz eingeben

Zu beachten ist, dass die Eingaben (z.B. richtiges Datum) überprüft werden. Mit dem grünen Plus-Icon (Rechts-Oben) kann der nächste Datensatz hinzugefügt werden. Mit dem Disketten-Icon (Links-Oben) erfolgt ein COMMIT. Möglicherweise kann es zu Constraint-Verstößen kommen. Die employeeid muss neu sein, die reportsto muss dagegen auf einer vorhandenen employeeid entsprechen.

Abfragen

Icon 'Abfrage eingeben'. z.B.
Alle Namen mit P am Anfang: P%
Alle Namen mit dem ersten Buchstaben ab P: >P
Icon 'Abfrage ausführen'.

Datensatz ändern

Datensatz abfragen und ändern. Mit dem Disketten-Icon (Links-Oben) erfolgt ein Commit.

ORDER BY

Beispiel:
Sortierung nach LASTNAME
Im Forms Builder im Objektnavigator den Datenblock EMPLOYEES doppelklicken damit die Attributpalette erscheint.
In der Attribut-Palette 'ORDER BY' suchen. Dazu kann die Suchfunktion (Rechts-Oben) verwendet werden.
Hinter 'ORDER BY' LASTNAME, FIRSTNAME eintragen.
Speichern und Form ausführen.

4.3.4 Master-/Detail-Formular

Das Master-/Detail-Formular besteht aus zwei Blöcken, um zwei verknüpfte Tabellen bearbeiten zu können.

Menü Werkzeuge – Datenblock-Assistent – Tabelle oder Ansicht.
'Durchsuchen' anklicken. Es erscheinen die Tabellen der Datenbank.
Auswahl 'EMPLOYEES'. Es erscheinen die Spalten der Tabelle.
Tabellenblockname: EMPLOYEETERRITORIES
Datenintentigrität erzwingen.
Beziehung erstellen: Auswahl EMPLOYEES
Datenblöcke automatisch verknüpfen – aktivieren
Detail-Objekt: EMPLOYEEID
Master-Objekt: EMPLOYEEID
Datenblockname: EMPLOYEETERRITORIES
Datenblock erstellen und danach Layoutmanager aufrufen.
...
Rahmentitel: employeeterritories

Es erscheint der Layout-Manager mit den beiden Formularen.

4.3.5 Ändern der Aktivierungsreihenfolge

Die Reihenfolge des Aktivieren der Felder (Tab-Taste) kann unabhängig von der Reihenfolge ihrer Anzeige geändert werden.
Wechseln zum Objektnavigator.
Aufklappen von Forms – NORTHWIND_FORMULAR01 – Datenblöcke – EMPLOYEES
Ziehen eines Spaltennamens (z.B. TITLE) an die erste Stelle.

4.3.6 Master-/Detail-Formular für Inner-Join

Die Spalte REPORTSTO zeigt auf die EMPLOYEEID des jeweiligen Chefs.
Menü Werkzeuge – Datenblock-Assistent – Tabelle oder Ansicht.
'Durchsuchen' anklicken. Es erscheinen die Tabellen der Datenbank.
Auswahl 'EMPLOYEES'. Es erscheinen die Spalten der Tabelle.
Auswahl der Spalten EMPLOYID, TITLE, LASTNAME, FIRSTNAME
Tabellenblockname: EMPLOYEES
Datenintentigrität erzwingen.
Beziehung erstellen: Auswahl EMPLOYEES
Aus JOIN-Bedingung basierend
Detail-Objekt: EMPLOYEEID

Master-Objekt: REPORTSTO
Datenblockname: BOSS
Datenblock erstellen und danach Layoutmanager aufrufen.
...
Es erscheint der Layout-Manager mit den Formularen.
Auswahl der Spalten TITLE, LASTNAME, FIRSTNAME
Rahmentitel: BOSS
Beachte: Im Layout-Manager müssen alle Formulare auf die Leinwand passen, sonst werden diese nicht angezeigt.

4.3.7 Wertelisten (LOV)

Beispiel:
Beschränkung der Anzeige von REPORTSTO auf vorhandene EMPLOYEEIDs.
Im Objektnavigator auf Wertelisten (LOV) und danach das Pluszeichen anklicken. Wertelisten-Assitent verwenden.
Neue Datensatzgruppe
SQL-Abfrage:

```
select EMPLOYEEID as REPORTSTO from employees order by EMPLOYEEID
```

EMPLOYEEID auswählen.

Titel: REPORTSTO
...
Fertig. Unter Wertelisten und Datensatzgruppen erscheint je ein neues Objekt.

Zuweisung der Werteliste zu REPORTSTO.
Auswahl der Spalte REPORTSTO in der Tabelle EMPLOYEES.
Hinter Werteliste in der Attributpalette die neu erstellte Werteliste auswählen.
Aus Liste validieren: Ja

Im Objektnavigator unter Wertelisten (LOV) die neu erstellte Werteliste auswählen und in der Attributpalette folgende Änderungen durchführen.
Attribute Spaltenzuordnung: Weitere Optionen
Rückgabeobjekt: EMPLOYEES.REPORTSTO
Eventuel unter 'Physikalisch' die Höhe und Breite der Dialogbox ändern.

Menü Datei - Speichern
Testen des Formulars
Ctrl + L – Anzeige der Werteliste im Feld REPORTSTO (Hinweis dazu in der Statuszeile).

Anhang A

Appendix

A.1 SQL/PL – Übungen und Lösungen

A.1.1 Einfache SELECT-Abfragen

Schreiben Sie eine Abfrage, die alle Datensätze der Tabelle products von Lieferanten mit der Nummer 1 liefert.

```
-- Lösung:
select * from products where SUPPLIERID=1;
```

Schreiben Sie eine Abfrage, die alle Datensätze der Tabelle products mit der Lieferanten_Nummer = 1 und Kategorie_Nummer = 2 liefert.

```
-- Lösung:
select * from products where SUPPLIERID=2 and CATEGORYID=1;
```

Schreiben Sie eine Abfrage, die alle Datensätze der Tabelle products mit Preisen >= 50 und <= 100 anzeigt.

```
-- Lösung:
select * from products where UNITPRICE between 50 and 100;
```

Schreiben Sie eine Abfrage, die alle Datensätze der Tabelle products mit Netto- und Bruttopreisen anzeigt.

```
-- Lösung:
select UNITPRICE as Netto, UNITPRICE*1.16 as Brutto from products;
```

Schreiben Sie eine Abfrage, die alle Datensätze der Tabelle employees mit Vor- und Nachname, getrennt durch ein Lerrzeichen anzeigt.

```
-- Lösung:
select FIRSTNAME || ' ' ||  LASTNAME from employees;
```

Schreiben Sie eine Abfrage, die alle Datensätze der Tabelle employees mit Mitarbeitern aus London, Redmond oder Seattle liefert.

```
-- Lösung:
select LASTNAME from employees
  where CITY in ('London','Redmond','Seattle')
  order by 1
;
```

Schreiben Sie eine Abfrage, die die Nachnamen aller Mitarbeiter aus den USA anzeigt. Die Nachnamen sollen absteigend sortiert werden.

```
-- Lösung:
select LASTNAME from employees where COUNTRY != 'USA' order by 1 desc;
```

Schreiben Sie eine Abfrage, die alle Produktnamen mit deren Kategorie-ID auflistet. Dabei soll jeweils folgender Satz ausgegeben werden:
Das Produkt X gehört zur Kategorie Y.

```
-- Lösung:
select
  'Das Produkt' || PRODUCTNAME || ' gehört zur Kategorie ' || CATEGORYID || '. '
  from products;
```

A.1.2 LIKE, Platzhalter

Erstellen Sie eine Abfrage, die Ihnen alle Produkte ausgibt, die mit Cänfangen. Diese sollen absteigen nach dem Produktname sortiert sein.

```
-- Lösung:
select PRODUCTNAME from products
  where PRODUCTNAME
  like 'C%'
  order by PRODUCTNAME desc
;
```

Erstellen Sie eine Abfrage, die Ihnen alle Produkte ausgibt, die mit Ä“, ”B“, oder Cänfangen.

```
-- Lösung:
select PRODUCTNAME from products
  where PRODUCTNAME
  like 'A%' or
    PRODUCTNAME like 'B%' or
    PRODUCTNAME like 'C%'
;
```

Erstellen Sie eine Abfrage, die Ihnen alle Produkte ausgibt, bei denen der 5. Buchstabe ein cöder “dïst. Diese sollen aufsteigend nach dem Produktname sortiert sein.

```
-- Lösung:
select PRODUCTNAME from products where PRODUCTNAME
  like '____c%' or PRODUCTNAME like '____d%' order by 1
;
```

Erstellen Sie eine Abfrage, die Ihnen alle Produkte ausgibt, bei denen der 5. Buchstabe kein cïst.

```
-- Lösung:
select PRODUCTNAME from products where PRODUCTNAME not
  like '____c%'
;
```

A.1.3 Gruppierungen

Schreiben Sie eine Abfrage, die Ihnen die Spalte City der Tabelle employees ausgibt. Es soll keine Stadt mehrmals auftauchen.

```
CITY
----------------
Seattle
Tacoma
Ta
-- Lösung:
select distinct city from employees;
```

Schreiben Sie eine Abfrage, die Ihnen die Summe des Warenbestandes (unitsinstock) aus der Tabelle products, gruppiert nach den Lieferantennummern (supplierid) ausgibt.

```
SUPPLIERID SUMME
---------------
1          69
-- Lösung:
select supplierid,sum(unitsinstock) as Summe from products
  group by supplierid
;
```

Schreiben Sie eine Abfrage, die Ihnen die Summe des Warenbestandes (unitsinstock) aus der Tabelle products, gruppiert nach den Lieferantennummern (supplierid) ausgibt! Es sollen nur die Lieferanten erscheinen, bei denen die Summe des warenbestandes größer als 70 ist.

```
SUPPLIERID SUMME
---------------
2          133
-- Lösung:
select supplierid,sum(unitsinstock) as Summe from products
  group by supplierid having sum(unitsinstock) > 70
;
```

Schreiben Sie eine Abfrage, die Ihnen die Summe des Warenbestandes (unitsinstock) aus der Tabelle products, gruppiert nach den Lieferantennummern (supplierid) und den Kategorienummern (categoryid) ausgibt!

```
SUPPLIERID CATEGORYID  SUM(UNITSINSTOCK)
----------------------------------------
         1          1                56
         1          2                13
         2          2               133
         3          2               126
-- Lösung:
select supplierid,categoryid,sum(unitsinstock) from products
  group by supplierid,categoryid
;
```

Listen Sie alle Lieferanten auf, die mehr als 5 Produkte liefern!

```
SUPPLIERID
----------
         7
        12
-- Lösung:
select supplierid from products
  group by supplierid having count(Productname) > 5
;
```

Listen Sie die Lieferanten mit Ihren jeweils teuersten Produkt auf. Es sollen nur die Lieferanten angezeigt werden, bei denen das teuerste Produkt billiger als 10 DM ist.

```
SUPPLIERID  MAX(UNITPRICE)
--------------------------
        10            4,5
-- Lösung:
select supplierid,max(unitprice) from products
  group by supplierid having max(unitprice) < 10
;
```

Zusatzfrage:
Listen Sie den Nachnamen des Mitarbeiters (employees) auf, der am längsten angestellt war!

```
-- Lösung:
SELECT lastname,hiredate FROM employees
  WHERE hiredate in
   (
     SELECT MIN(hiredate) FROM employees
   )
;
```

Was machen folgende Abfragen?

```
select lastname from employees group by city;
-- Lösung:
-- Fehler -- Falsche Gruppierung.

select lastname from employees where city not in('Berlin','London');
-- Lösung:
-- Anzeige aller Angestellten, die nicht aus Berlin oder London kommen.

select discontinued,sum(unitprice*unitsinstock) from products group by discontinued;
-- Lösung:
-- Anzeige des Warenbestandes im Warenlager.

select supplierid,sum(unitprice*unitsinstock) from products group by supplierid;
-- Lösung:
-- Werte der Produkte jedes Lieferanten im Warenlager.

select categoryid,sum(unitprice*unitsinstock) from products group by supplierid;
-- Lösung:
-- Fehler -- Falsche Gruppierung.

select sum(unitprice*unitsinstock) from products group by supplierid;
-- Lösung:
-- Syntaktisch richtig aber unlogisch (sum).

select sum(unitprice*unitsinstock) from products;
-- Lösung:
-- Anzeige des Warenbestandes im Warenlager.
```

A.1.4 JOIN

Schreiben Sie eine Abfrage, die Ihnen die Kontaktnamen der Lieferanten ausgibt, die Produkte der Kategorie 1 oder 2 oder 3 liefern!

```
CONTACTNAME                    PRODUCTNAME
---------------------------------------
Charlotte Cooper               Chai
Charlotte Cooper               Chang
Charlotte Cooper               Aniseed Syrup
Shelley Burke                  Chef Anton's Cajun Seasoning
-- Lösung:
select s.CONTACTNAME,p.PRODUCTNAME from products p,suppliers s
  where p.CATEGORYID in ('1','2','3')
;
```

Schreiben Sie eine Abfrage, die Ihnen die Kundennamen, die entsprechenden Bestellungnummern (orderid) und den Namen des Angestellten (employees), der die Bestellung bearbeitet hat, ausgibt. Es soll nach dem Contactname geordnet sein!

```
CONTACTNAME                    ORDERID    LASTNAME
-------------------------------------------------
Alejandra Camino               10281      Peacock
Alejandra Camino               10282      Peacock
Alejandra Camino               10306      Davolio
Alejandra Camino               10917      Peacock

-- Lösung:
select c.CONTACTNAME,o.orderid,e.lastname from customers c, orders o, employees e
  where e.employeeID = o.employeeID and o.customerid = c.customerid
;
```

Schreiben Sie eine Abfrage, die Ihnen alle Kundennamen ausgibt, die eine Bestellung aufgegeben haben, die Produkte beinhaltet, die von einem Lieferanten aus Berlin geliefert werden. Es soll absteigend nach den Kundennamen sortiert werden!

```
-- Lösung:
select c.contactname
  from
    customers c,orders o,order_details od,products p,suppliers s
  where
    o.customerid = c.customerid and
    od.orderid = o.orderid and
    od.productid = p.productid and
    p.supplierid = s.supplierid and
    s.city='Berlin'
  order by 1
;
```

Schreiben Sie eine Abfrage, die Ihnen alle Kundennamen ausgibt, die eine Bestellung aufgegeben haben, die von Speedy Express ausgeliefert wurde!

```
-- Lösung:
select distinct c.COMPANYNAME from  customers c,  orders o,  shippers sh
  where
    c.customerid = o.customerid and
    o.SHIPVIA = sh.shipperid and
    sh.COMPANYNAME = 'Speedy Express'
;
```

Schreiben Sie eine Abfrage, die Ihnen die Städte ausgibt, in denen Kunden wohnen und die Städte darunter ausgibt, in denen Lieferanten wohnen. Jede Stadt soll nun einmal auftauchen!

```
CITY
-------
Aachen
Albuquerque
Anchorage
Ann Arbor
-- Lösung:
select distinct city from customers
  union
select distinct city from suppliers;
```

Schreiben Sie eine Abfrage, die Ihnen die Städte ausgibt, in den sowohl Kunden als auch Lieferanten wohnen. Jede Stadt soll nur einmal auftauchen!

```
CITY
-------
Berlin
London
-- Lösung:
select distinct city from customers
  intersect
select distinct city from suppliers;
```

Schreiben Sie eine Abfrage, die Ihnen die Städte ausgibt, in den Kunden, nicht aber Lieferanten, wohnen. Jede Stadt soll nun einmal auftauchen!

```
CITY
------
Aachen
Albuquerque
-- Lösung:
select distinct city from customers
  minus
select distinct city from suppliers;
```

Schreiben Sie eine Abfrage, die Ihnen die Kundennamen ausgibt, die noch nie etwas bestellt haben!

```
-- Lösung:
select distinct COMPANYNAME from customers
  minus
select distinct COMPANYNAME from customers c,orders o
  where c.customerid = o.customerid
;
```

Schreiben Sie eine Abfrage, die Ihnen die Lieferanten ausgibt, die keine Produkte liefern!

```
-- Lösung:
select distinct s.companyname from suppliers s,products p
  minus
select distinct s.companyname from suppliers s,products p
  where s.supplierid = p.supplierid
;
```

Schreiben Sie eine Abfrage, die Ihnen die Angestellten und Vorgesetzten ausgibt. Die reportsto-Spalte gibt die employeeid des Vorgesetzten an, der wiederum auch selbst ein Angestellter ist!

```
-- Lösung:
select e.LASTNAME,e.FIRSTNAME, c.LASTNAME,c.FIRSTNAME from
employees e,employees c
  where e.reportsto = c.employeeid(+)
;
```

Schreiben Sie eine Abfrage, die Ihnen die Kunden und Ihren Gesamtbestellwert (ohne Discount) auflistet.

```
CUSTO SUM(UNITPRICE
---------------
ALFKI     9192,4
ANATR     2805,9
ANTON    15030,7
AROUT      27613
BERGS    53936,3
BLAUS     4763,6
-- Lösung:
select c.customerid, sum(od.unitprice*od.quantity)
  from customers c, orders o, order_details od
  where c.customerid = o.customerid
    and o.orderid = od.orderid
  group by c.customerid
;
```

Wieviel Produkte hat ALFKI bestellt?

```
-- Lösung:
select count(*) from orders o,order_details od
 where o.customerid = 'ALFKI' and o.orderid = od.orderid
;
```

Wie ist der Gesamtumsatz von ALFKI?

```
-- Lösung:
select to_char(round(sum(od.UNITPRICE * od.quantity),2),'9999D00') as Gesamtumsatz
 from orders o, order_details od
 where o.customerid = 'ALFKI' and o.orderid = od.orderid
;
```

Was machen folgende Abfragen?

```
select * from products pr,categories ca};
-- Lösung:
-- ??

select pr.productname, ca.categoryname
  from products rp,categories ca
  where pr.productid=ca.categoryid};
-- Lösung:
-- ??
```

```
select pr.productname, ca.categoryname
  from products pr,categories ca
  where pr.categoryid(+)=ca.categoryid};
-- Lösung:
-- ??

select sum(pr.unitprice*pr.unitsinstock), ca.categoryname
  from products pr,categories ca
  where pr.productid=ca.categoryid group by ca.categoryname};
-- Lösung:
-- ??

select sum(pr.unitprice*pr.unitsinstock), ca.categoryname
  from products pr,categories ca group by ca.categoryname
  where pr.productid=ca.categoryid};
-- Lösung:
-- ??
```

A.1.5 Verschachtelte Abfragen

Wie heißt der älteste Mitarbeiter?

```
-- Lösung:
select lastname,firstname,birthdate
  from employees where birthdate in
  (select min(birthdate) from employees)
;
-- Peacock
```

Wie heißt das teuerste Produkt eines Lieferanten aus "Berlin"?

```
-- Lösung:
select PRODUCTNAME,unitprice from products p,suppliers s
  where s.supplierid = p.supplierid and
  unitprice =
  (
   select max(unitprice) from products p2
    where s.supplierid = p2.supplierid and supplierid in
     (
      select supplierid from suppliers where upper(city) = 'BERLIN'
     )
  )
;
-- Schoggi Schokolade 43,90
```

Welche Produkte (productname) sind überdurchschnittlich teuer?

```
-- Lösung:
select productname,UNITPRICE from products
  where UNITPRICE > (select avg(UNITPRICE) from products);
```

Wie heißt der jüngste Mitarbeiter?

```
-- Lösung:
select lastname,firstname,birthdate from employees
  where birthdate in (select max(birthdate) from employees);
-- Dadworth ??
```

Zu welcher Kategorie gehört das teuerste Produkt (categoryname)

```
-- Lösung:
select CATEGORYID,unitprice from products
  where unitprice =
   (
    select max(unitprice) from products
   )
;
```

Schreiben Sie eine Abfrage, die Ihnen die Lieferanten (Companyname) und daneben das teuerste Produkt dieses Lieferanten (Productname) und daneben den Preis ausgibt.

```
-- Lösung:
SELECT s.companyname, p.productname, p.unitprice
  FROM suppliers s, products p
  WHERE p.supplierid = s.supplierid AND p.unitprice =
   (
    SELECT MAX(p2.unitprice)
      FROM products p2, suppliers s2
      WHERE p2.supplierid = s2.supplierid
        AND s2.supplierid=s.supplierid
   )
;
```

Welcher Kunde hat bisher am meisten bestellt?

```
-- Lösung:
select companyname, UMSATZ from
  (
    select c.companyname, sum(od.unitprice*od.quantity)
      as Umsatz from orders o,order_details od, customers c
      where o.orderid = od.orderid and
         c.customerid = o.customerid group by c.companyname
  )
  where UMSATZ =
   (
     select max(UMSATZ) from
      (
        select c.companyname, sum(od.unitprice*od.quantity) as Umsatz
          from orders o,order_details od, customers c
          where o.orderid = od.orderid and
                c.customerid = o.customerid
          group by c.companyname
      )
   )
;
```

Erstellen Sie eine Abfrage, die Ihnen die Kunden (contactname) und die bisherigen Umsätze auflistet.

```
-- Lösung:
select c.contactname, sum(od.unitprice*od.quantity) as Umsatz
   from orders o,order_details od, customers c
   where o.orderid = od.orderid and
         c.customerid = o.customerid
   group by c.contactname
;
```

Welche Kunden haben bisher Chai bestellt?

```
-- Lösung Variante 1:
SELECT DISTINCT o.customerid
  FROM orders o, order_details od, products p
  WHERE o.orderid = od.orderid
    AND od.productid = p.productid
    AND UPPER(p.productname) = 'CHAI'
;
-- Lösung Variante 2:
select distinct companyname from customers
  where customerid in
   (
     select customerid from orders where orderid in
      (
       select orderid from order_details where PRODUCTID =
        (
          select PRODUCTID from products where PRODUCTNAME = 'Chai'
        )
      )
   )
;
```

Wie heißt der zweitälteste Mitarbeiter?

```
-- Lösung Variante 1:
select lastname,firstname,birthdate from employees
  where birthdate in
   (
     select min(birthdate) from employees where birthdate >
       (
         select min(birthdate) from employees
       )
   )
;
-- Lösung Variante 2:
select lastname from employees where birthdate=
(
  select min(birthdate)
  from
  (
    select lastname, birthdate from employees
    minus
    select lastname, birthdate from employees
      where birthdate = (select min(birthdate) from employees)
  )
);
```

Wie heißt der drittjüngste Mitarbeiter?

```
-- Lösung:
select lastname,firstname,birthdate from employees
  where birthdate =
   (
     select max(birthdate) from employees where birthdate <
       (
         select max(birthdate) from employees where birthdate <
           (
              select max(birthdate) from employees
           )
       )
   )
;
```

A.1.6 Funktionen

Schreiben Sie eine Abfrage, die Ihnen alle lastname der Employees in Großbuchstaben ausgibt.

```
-- Lösung:
select upper(lastname) from employees;
```

Schreiben Sie eine Abfrage, die Ihnen alle lastnames der Employees in Kleinbuchstaben ausgibt.

```
-- Lösung:
select lower(lastname) from employees;
```

Schreiben Sie eine Abfrage, die Ihnen den ersten Buchstaben von lastnames der Employees in Großbuchstaben und den Rest in Kleinbuchstaben ausgibt.

```
-- Lösung:
select initcap(lastname) from employees;
```

Schreiben Sie eine Abfrage, die Ihnen den ersten und den letzten Buchstaben von lastnames der Employees in Großbuchstaben und den Rest in Kleinbuchstaben ausgibt.

```
-- Lösung:
select initcap(substr(lastname,1,length(lastname)-1)) ||
    upper(substr(lastname,-1,1))
  from employees;
```

Schreiben Sie eine Abfrage, die Ihnen alle Employees ausgibt, in deren lastname zwei l (also ll) vorkommen. Es soll dabei egal sein, ob es jeweils ein großes oder kleines l sind.

```
-- Lösung:
select lastname from employees where instr(lower(LASTNAME),'ll') > 0;
```

Schreiben Sie eine Abfrage, die Ihnen eine Spalte als Ergebnismenge zurückgibt, in welcher der erste Buchstaben des Vornamens, dahinter ein Punkt und dahinter, getrennt durch ein Leerzeichen, der lastname ausgegeben wird. Bei dem lastname sollte lediglich der erste Buchstabe groß sein. (z.B. R. King)

```
-- Lösung:
select upper(substr(firstname,1,1)) || '. ' || initcap(lastname)
  from employees;
```

Beantworten Sie folgende Prüfungsfrage:
Which SELECT statement will get the result 'elloworld' fromt the string 'Hello World'?

```
SELECT SUBSTR ('HelloWorld',1) FROM dual;
SELECT INITCAP(TRIM('HellowWorld', 1,1) FROM dual;
SELECT LOWER(SUBSTR ('HellowWorld', 2,1) FROM dual;
SELECT LOWER(SUBSTR('HellowWorld', 2,1) FROM dual;
SELECT LOWER(TRIM ('H' FROM 'HelloWorld')) FROM dual;
-- Lösung:
Die letzte Abfrage ist richtig.
```

Ermitteln Sie alle Sonntagskinder aus der Tabelle employees. Zurückgegeben werden soll der lastname und das Geburtsjahr.

```
-- Lösung:
select lastname, birthdate, to_char(birthdate,'fmDAY')
  from employees where to_char(birthdate,'fmDAY') = 'SONNTAG'
;
```

Ermitteln Sie das Datum, an dem die employees ihr 25-jähriges Firmenzugehörigkeitsjubiläum feiern können. Folgendermaßen soll das Eregbnis dargestellt werden:

```
LASTNAME             Jubilee
-------------------- -----------------
Davolio                      01.Mai       2017
Fuller                       14.August    2017
Leverling                    01.April     2017
Peacock                      03.Mai       2018
Buchanan                     17.Oktober   2018
Suyama                       17.Oktober   2018
-- Lösung:
select LASTNAME,
       to_char(add_months(hiredate,25*12),'dd.Month     YYYY')
         as Jubilee
  from employees
;
```

Ermitteln Sie den Mitarbeiter, der am längsten angestellt war.

```
-- Lösung:
select lastname, hiredate from employees
  where hiredate in
   (select min(hiredate) from employees)
;
```

Ermitteln Sie den Mitarbeiter, der am ältesten war, als er eingestellt wurde.

```
-- Lösung:
select lastname, birthdate,hiredate, months_between(hiredate,birthdate)
  from employees
  where months_between(hiredate,birthdate) =
   (select max(months_between(hiredate,birthdate)) from employees)
;
```

Ermitteln Sie die Mitarbeiter, die gegenwärtig länger als 10 Jahre in der Firma arbeiten

```
-- Lösung:
select lastname, hiredate, months_between(sysdate,hiredate)/12 as years
  from employees
  where months_between(sysdate,hiredate)/12 > 10
;
```

Schreiben Sie ein Abfrage, die den Produktpreis mit jeweils zwei Nachkommastellen ausgibt.

```
-- Lösung:
select productname,to_char(unitprice,'99999D00') from products;
```

A.1.7 CREATE, INSERT, MODIFY

Erstellen Sie folgende Tabellen (Legen Sie alle Anweisungen als Skript in einer Datei ab).

```
-- Tabelle Lehrer
Anrede       Name    Vorname
------------------
Frau Janker  Hanni
Herr Hubert  Jan
Frau Hinze   Jutta
Herr Hub     Frank
Herr Kreis   Klaus
-- Lösung:
create table lehrer (
 anrede  varchar(40) not null,
 name    varchar(40) not null,
 vorname varchar(40) not null
);
insert into lehrer values ('Frau','Janker','Hanni');
insert into lehrer values ('Herr','Hubert','Jan');
insert into lehrer values ('Frau','Hinze','Jutta');
insert into lehrer values ('Herr','Hub','Frank');
insert into lehrer values ('Herr','Kreis','Klaus');
select * from lehrer;
```

Fügen Sie zur Tabelle eine weitere Spalte namens Gehalt (Typ: int) hinzu.

```
-- Lösung:
alter table lehrer add (gehalt int);
```

Vergrößern Sie die Länge des Datentypes für die Spalte Name um 5.

```
-- Lösung:
alter table lehrer modify (name varchar(45));
```

Setzen Sie das Gehalt für alle Herren auf 3000.

```
-- Lösung:
UPDATE lehrer SET gehalt = 3000 where ANREDE = 'Herr';
commit;
```

Setzen Sie das Gehalt für alle Frauen auf 3100.

```
-- Lösung:
UPDATE lehrer SET gehalt = 3100 where ANREDE = 'Frau';
commit;
```

Erstellen und füllen Sie eine Tabelle namens Herren (Name, Vorname, Gehalt).

```
-- Lösung:
create table herren as select name,vorname,gehalt from lehrer where ANREDE = 'Herr';
```

Löschen Sie aus der Tabelle Lehrer alle Herren.

```
-- Lösung:
delete from lehrer where ANREDE = 'Herr';
commit;
```

Benennen Sie die Tabelle Lehrer in Frauen um.

```
-- Lösung:
rename lehrer to frauen;
```

A.1.8 Contraints

Erstellen Sie alle Beziehung der Northwind-Datenbank. Speichern Sie die Anweisungen in einer Datei!

```
-- Lösung:
---Primary Keys:
ALTER TABLE employees
  ADD CONSTRAINT empl_pk PRIMARY KEY (employeeid);
ALTER TABLE employeeterritories
  ADD CONSTRAINT emplterr_pk PRIMARY KEY (employeeid,territoryid);
ALTER TABLE territories
  ADD CONSTRAINT terr_pk PRIMARY KEY (territoryid);
ALTER TABLE region
  ADD CONSTRAINT region_pk PRIMARY KEY (regionid);
ALTER TABLE customers
  ADD CONSTRAINT customers_pk PRIMARY KEY (customerid);
ALTER TABLE orders
  ADD CONSTRAINT orders_pk PRIMARY KEY (orderid);
ALTER TABLE order_details
  ADD CONSTRAINT order_details_pk PRIMARY KEY (orderid,productid);
ALTER TABLE products
```

```
  ADD CONSTRAINT products_pk PRIMARY KEY (productid);
ALTER TABLE categories
  ADD CONSTRAINT categories_pk PRIMARY KEY (categoryid);
ALTER TABLE suppliers
  ADD CONSTRAINT suppliers_pk PRIMARY KEY (supplierid);
ALTER TABLE shippers
  ADD CONSTRAINT shippers_pk PRIMARY KEY (shipperid);

-- Foreign Keys:
ALTER TABLE products
  ADD CONSTRAINT prod_fk_cat
  FOREIGN KEY (categoryid)
  REFERENCES categories(categoryid);
ALTER TABLE products
  ADD CONSTRAINT prod_fk_supp
  FOREIGN KEY (supplierid)
  REFERENCES suppliers(supplierid);
ALTER TABLE order_details
  ADD CONSTRAINT orderd_fk_prod
  FOREIGN KEY (productid)
  REFERENCES products(productid);
ALTER TABLE order_details
  ADD CONSTRAINT orderd_fk_orders
  FOREIGN KEY (orderid)
  REFERENCES orders(orderid);
ALTER TABLE orders
  ADD CONSTRAINT orders_fk_shippers
  FOREIGN KEY (shipvia)
  REFERENCES shippers(shipperid);
ALTER TABLE orders
  ADD CONSTRAINT orders_fk_customers
  FOREIGN KEY (customerid)
  REFERENCES customers(customerid);
ALTER TABLE employeeterritories
  ADD CONSTRAINT emplterr_fk_employ
  FOREIGN KEY (employeeid)
  REFERENCES employees(employeeid);
ALTER TABLE employeeterritories
  ADD CONSTRAINT emplterr_fk_terr
  FOREIGN KEY (territoryid)
  REFERENCES territories(territoryid);
ALTER TABLE territories
  ADD CONSTRAINT terr_fk_region
  FOREIGN KEY (regionid)
  REFERENCES region(regionid);
ALTER TABLE employees
  ADD CONSTRAINT employ_fk_employ
  FOREIGN KEY (reportsto)
  REFERENCES employees(employeeid);
```

Stellen Sie folgendes sicher (Speichern Sie auch diese Anweisungen in einer Datei): Der unitprice muss größer oder gleich 0 sein (products und order_details).

```
-- Lösung:
ALTER TABLE products ADD CONSTRAINT prod_1_CK1 CHECK (unitprice >= 0);
ALTER TABLE order_details ADD CONSTRAINT orderd_1_CK1 CHECK (unitprice >= 0);
```

Das Land der Kunden darf nicht Irak oder Nordkorea oder Kuba sein.

```
-- Lösung:
ALTER TABLE customers
  ADD CONSTRAINT custmom_CK1
    CHECK (lower(city) NOT IN ('iraq','north korea','north corea','cuba'))
;
```

Das Shippeddate darf nicht vor dem Orderdate liegen (Tabelle orders).

```
-- Lösung:
ALTER TABLE orders
  ADD CONSTRAINT orders2ship_CK1
    CHECK (shippeddate >= orderdate)
;
```

Das Requireddate darf nicht vor dem Orderdate liegen (Tabelle orders).

```
-- Lösung:
ALTER TABLE orders
  ADD CONSTRAINT req2order_CK1
    CHECK (requireddate >= orderdate)
;
```

Kontrolle:

```
select
    TABLE_NAME,
    CONSTRAINT_NAME,
    SEARCH_CONDITION,
    R_CONSTRAINT_NAME,
    INDEX_NAME
  from user_constraints
  where user = 'SYS'
    and last_change > sysdate-1
;
```

Abfrage: Liefert mehrfach vorkommende Werte in employees:

```
-- Lösung:
select EMPLOYEEID from employees minus select distinct EMPLOYEEID from employees;
select customerid from orders minus select distinct customerid from orders;
```

A.1.9 PL/SQL

Erstellen Sie eine Prozedur New_Empl, mit der Sie in die Tabelle employees neue Mitarbeiter hinzufügen können. Die Mitarbeiternummer des neuen Mitarbeiters muss um 1 größer sein als die größte Mitarbeiternummer. Aufgerufen werden soll die Prozedur mit exec new_empl('Schroeder','Gerd').

```
-- Lösung:
/*
set serveroutput on;
create or replace procedure New_Empl(nachname varchar2,vorname varchar2) is
current_table      varchar2(30) := 'sys.employees';
```

```
max_employeeid     int          := 0;
current_employeeid int          := 0;
begin
   select max(employeeid) into max_employeeid from employees;
   current_employeeid := max_employeeid+1;
   insert into employees (employeeid,lastname,firstname)
      values (current_employeeid,nachname,vorname);
   commit;
           Exception
   when others then
          dbms_output.put_line('Error');
end;
/
show errors;
exec new_empl('Schroeder','Gerd')
```

Erstellen Sie eine Prozedur Prim_Int, welche die Primzahlen zwischen in einem bestimmten Intervall ausgibt. Aufgerufen werden soll die Prozedur mit
`exec Prim_Int(10,20)`
Hilfe: `select mod(11,4) from dual;`

```
-- Lösung Variante 1:
create or replace procedure Prim_INT(unten int, oben int) is
temp int:=1;
begin
  If (oben < unten) OR (oben < 1) OR (unten < 1) then
      dbms_output.put_line('witzig');
    else
      for x in unten .. oben loop
        temp:=1;
          for y in 2 .. (x-1) loop
            If mod(x,y) = 0 then
              temp:=0;
            End If;
          end loop;
          If temp=1 then
            dbms_output.put_line(x);
          End If;
      end loop;
  end if;
  Exception
    when others then
      dbms_output.put_line('unbekannte Ausnahmeverletzung');
end;
/
show errors;
exec Prim_Int(5,10);

-- Lösung Variante 2:
set serveroutput on;
create or replace procedure Prim_Int(first_num int,sec_num int) is
zahl number(5):=0;
n number(5):=0;
prim number(5):=1;
begin
  dbms_output.put_line('.');
  dbms_output.put_line('Zwischen '||first_num||' und '||sec_num||' gibt es folgende Primzahlen:');
```

```
  for x in first_num .. sec_num loop
    for n in 2 .. (trunc(x/2)) loop
      select mod(x,n) into zahl from dual;
      if zahl > 0
        then
          prim:=1;
          zahl:=0;
        else
          prim:=0;
      end if;
    end loop;
    if prim = 1
      then dbms_output.put_line(x);
    end if;
  end loop;
  Exception
    when too_many_rows then
      dbms_output.put_line('Sie haben zu viele Werte');
    when others then
      dbms_output.put_line('unbekannte Ausnahmeverletzung');
end;
/
show errors;
```

Zusatzaufgabe 1:
Wandeln Sie die Prozedur Prim_Int um, indem Sie andere Schleifenkonstrukte verwenden.

```
-- Lösung:
??
```

Zusatzaufgabe 2:
Verbessern Sie die Gültigkeitsprüfung bei der Parameterübergabe.

```
-- Lösung:
??
```

A.1.10 Neues von Oracle 9i

Welche Kunden (contactname) haben Produkte der Kategorie beverages bestellt. Benutzen Sie NATURAL JOINs, soweit es möglich ist.

```
-- Lösung:
select distinct contactname from customers
  NATURAL JOIN orders
  NATURAL JOIN order_details
  NATURAL JOIN products
  NATURAL JOIN categories
   where lower(categoryname) = 'beverages'
;
```

Welcher Kunde hat noch nie etwas bestellt? Benutzen Sie einen OUTER JOIN.

```
-- Lösung:
select distinct contactname, orderid from customers c
  left join orders o on c.customerid = o.customerid
    where orderid is null
;
```

Welche Kunden wurden bisher von Speedy Express beliefert? Benutzen Sie INNER JOIN und NATURAL JOIN, soweit es möglich ist.

```
-- Lösung:
create view shippers_view as
  select shipperid as shipvia, companyname as sh_companyname from shippers
;
select distinct companyname, sh_companyname from customers
  NATURAL JOIN orders
  NATURAL JOIN shippers_view
    where lower(sh_companyname) = 'speedy express'
;
drop view shippers_view
;
```

Erstellen Sie eine Abfrage, die Ihnen rechts den Angestellten und links den Vorgesetzten ausgibt. Der Chef sollte in der Spalte Vorgesetzter BOSS zu stehen haben (Tipp: NVL).

```
-- Lösung:
select e.lastname Mitarbeiter,nvl(boss.lastname,'Boss') as Chef
  from employees e
  left join employees boss on e.reportsto = boss.employeeid
;
```

Erstellen Sie eine Abfrage, welche die Durchschnittskommission von scott.emp ermittelt. Ein Null-Wert sollte hierbei als 0 interpretiert werden und demnach in der Durchschnittsberechnung mit einbezogen werden

```
-- Lösung:
select avg(nvl(COMM,0)) from scott.emp;
```

Schreiben Sie eine Abfrage, welche die Mehrwertsteuer der Produkte berechnet. Produkte der Kategorie 1 haben die Mehrwertsteuer 0%, der Kategorie 2 und 3 Mehrwertsteuer 7% und der Rest 16%.

```
-- Lösung Variante 1:
SELECT productname,
  CASE
    WHEN p.categoryid = 1 THEN 0
    WHEN p.categoryid IN (2,3) THEN unitprice*0.07
    ELSE unitprice* 0.16
  END AS MwSt
  FROM products p
;
-- Lösung Variante 2:
select productname, to_char(unitprice, '999g999d99'),
  to_char(unitprice*
    (
```

```
      case categoryid
        when 1 then 0
        when 2 then 0.07
        when 3 then 0.07
        else 0.16
      end
    ), '999g999d99') as MWST
  from products natural join categories
;
```

A.1.11 Abschlussübung

Sie haben den Auftrag erhalten, für eine Schule eine Datenbank zu erstellen. Mit Hilfe dieser Datenbank soll es möglich sein, die Lehrer, Schüler, Räume, Fächer und Klassen zu verwalten. Es soll auch ein Stundenplan möglich sein. Weiterhin soll es auch möglich sein, einen Raumbelegungsplan einschließlich eines Stundenplanes für die Lehrer zu erstellen.

1. Überlegen Sie sich eine Struktur für diese Datenbank (2 Stunden) – mit anschließender Präsentation und Diskussion.
2. Setzen Sie diese Struktur in die Praxis um.
3. Falls Sie noch Zeit haben, schreiben Sie ein kleines Front-End mit einen Programm Ihrer Wahl.

```
-- Lösung:
create user schule identified by schule;
grant dba to schule;
connect schule/schule;
-- TABLEs
Create table Klasse (
  Klassen_id number(5) constraint kl_pk primary Key,
  Klassenname varchar2(10) not null);
Create table Fach (
  Fach_id number(5) constraint fa_pk primary Key,
  Fachname varchar2(10) not null,
  Fach_kurz varchar2(5));
Create table Schueler (
  Schueler_id number(5) constraint sch_pk primary Key,
  Nachname varchar2(20) not null,
  Vorname varchar2(20) not null,
  Geb_Datum date,
  Geschlecht varchar2(10) not null,
  Bemerkung varchar2(80),
  Klassen_id number(5) not null
    constraint kl_fk references klasse(klassen_id));
Create table Lehrer (
  Lehrer_id number(5)
    constraint le_pk primary Key,
  Nachname varchar2(20) not null,
  Vorname varchar2(20) not null,
  Geb_Datum date);
Create table Lehrer_Fach (
  Fach_Lehrer_id number(5) constraint le_fa_pk primary Key,
  Fach_id  number(5) not null
```

```
    constraint fa_fk references Fach(Fach_id),
  Lehrer_id number(5) not null
    constraint le_fk references Lehrer(Lehrer_id));
Create table Klassenraum (
  Raum_id number(5) constraint ra_pk primary Key,
  Raumbezeichnung varchar2(20) not null,
  Besonderes  varchar2(80));
Create table Wochentag (
  Tag_id number(5) constraint wo_pk primary Key,
  Tag varchar2(10) not null);
Create table Stunden (
  Stunden_id number(5) constraint std_pk primary Key,
  Stundennummer varchar2(10) not null);
Create table Stundenplan (
  Plan_id number(5) constraint srdpl_pk primary Key,
  Stunden_id number(5) not null
     constraint std_fk references Stunden(Stunden_id),
  Tag_id number(5) not null
    constraint ta_fk references Wochentag(Tag_id),
  Raum_id number(5) not null
    constraint ra_fk references Klassenraum(Raum_id),
  Klassen_id number(5) not null
    constraint kl2_fk references Klasse(Klassen_id),
  Fach_Lehrer_id number(5) not null
    constraint fa_le_fk references Lehrer_Fach(Fach_Lehrer_id)
);
commit;
-- CONSTRAINTs
ALTER  TABLE stundenplan ADD
  CONSTRAINT uk_T_S_L_sp UNIQUE (Tag_id,Stunden_id,Fach_lehrer_id);
ALTER  TABLE stundenplan ADD
  CONSTRAINT uk_T_S_R_sp UNIQUE (Tag_id,Stunden_id,Raum_id);
ALTER  TABLE stundenplan ADD
  CONSTRAINT uk_T_S_K_sp UNIQUE (Tag_id,Stunden_id,Klassen_id);
-- alter table stundenplan drop constraint uk_R_K_L_sp ;
-- views
-- Lehrer
create or replace view V_Lehrer as
  select l.nachname Nachname,
         w.tag Tag,
         s.stundennummer Stunde,
         f.Fachname Fach,
         r.raumbezeichnung Raum,
         k.klassenname
    from Stundenplan sp,
         Lehrer l,
         wochentag w,
         stunden s,
         fach f,
         klassenraum r,
         klasse k,
         Lehrer_fach lf
    where sp.fach_lehrer_id = lf.fach_lehrer_id and
         lf.lehrer_id = l.lehrer_id and
         sp.tag_id=w.tag_id and
         sp.stunden_id=s.stunden_id and
         lf.fach_id=f.fach_id and
         sp.raum_id=r.raum_id and
```

```
         sp.klassen_id=k.klassen_id
    order by l.nachname,w.tag_id,s.stunden_id;
-- Klassen
create or replace view V_Klasse as
  select k.klassenname,
         w.tag Tag,
         s.stundennummer Stunde,
         r.raumbezeichnung Raum,
         f.Fach_Kurz Fach,
         l.nachname lehrer
    from Stundenplan sp,
         Lehrer l,
         wochentag w,
         stunden s,
         fach f,
         klassenraum r,
         klasse k,
         Lehrer_fach lf
    where sp.fach_lehrer_id = lf.fach_lehrer_id and
         lf.lehrer_id=l.lehrer_id and
         sp.tag_id=w.tag_id and
         sp.stunden_id=s.stunden_id and
         lf.fach_id=f.fach_id and
         sp.raum_id=r.raum_id and
         sp.klassen_id=k.klassen_id
    order by k.klassenname,w.tag_id,s.stunden_id;
-- Raumbelegung
create or replace view V_Raum as
  select       r.raumbezeichnung Raum,
                       w.tag Tag,
                       s.stundennummer Stunde,
                       k.klassenname,
                       f.Fach_Kurz Fach,
                       l.nachname lehrer
        from Stundenplan sp,Lehrer l,wochentag w,stunden s,fach f,klassenraum r,klasse k,
                       Lehrer_fach lf
        where sp.fach_lehrer_id=lf.fach_lehrer_id and lf.lehrer_id=l.lehrer_id and
                sp.tag_id=w.tag_id and
                sp.stunden_id=s.stunden_id and
                lf.fach_id=f.fach_id and
                sp.raum_id=r.raum_id and
                sp.klassen_id=k.klassen_id
        order by r.raumbezeichnung,w.tag_id,s.stunden_id;
```

Oracle-Datenbank-Client mit M$-Access
Eine Bastellösung für Verspielte.

Ein Einloggen ist als 'sys as sysdba' ist nicht möglich. Deshalb ist ein entsprechender User anzulegen.
Access starten, Leere DB, OK, Dateiname vergeben (*.mdb) und speichern.
Menü Datei | Externe Daten | Tabellen verknüpfen, Dateityp: ODBC
Neu, Oracle in OraHome92, Weiter, Name der Datenquelle: testdb29, Fertigstellen
Username: schule
Password: schule
Servicename: testdb29
Tabellen verknüpfen

Tabelle SCHULE.STUNDENPLAN auswählen. OK
SCHULE.STUNDENPLAN in der Liste anklicken zum editieren.
Menü Datein | Speichern. (commit wird automatisch ausgeführt.)

A.2 Fundamentals I – Übungen und Lösungen

A.2.1 PFILE / SPFILE

In dieser Übung werden Sie die Abarbeitungsreihenfolge und -hierarchie der Initialisierungsparameterdateien kennen lernen. Bitte notieren Sie sich Ihre SID:

```
select name, value from v$parameter where name ='instance_name';
```

SID = testdb29

Legen Sie in Ihrem Home-Verzeichnis folgende Verzeichnisse an:

```
mkdir ~/spfile
mkdir ~/pfile
```

Im folgenden werden Sie ein PFILE aus dem standardmäßigen SPFILE erstellen.

1. Starten Sie den OEM. Melden Sie sich Standalone an.
2. Erweitern Sie 'Datenbanken'. Geben Sie in der Anmeldung an die Datenbank als Benutzer `sys` an. Das Kennwort ist auch `sys`. Bei 'Anmelden als' geben Sie das Privileg `SYSDBA` an.
3. Erweitern Sie 'Instanz' und wählen Sie 'Konfiguration' aus.
4. Wählen Sie im Menü Objekt – PFILE erstellen.
   Unter SPFILE geben Sie den Pfad zum Standard-SPFILE an. Der Dateiname soll `SPFILEtestdb29.ora` sein. Unter PFILE geben Sie den gleichen Pfad wie beim SPFILE an. Der Dateiname soll `inittestdb29.ora` sein.
5. Fahren Sie die Instanz herunter.
   Verschieben Sie die Datei `SPFILEtestdb29.ora` in das Verzeichnis `/spfile`.
   Achtung: Löschen Sie dieses SPFILE nicht!
6. Versuchen Sie nun Ihre Instanz über das SQL-Worksheet hochzufahren.

Was passiert und warum? Überprüfen Sie auch die Konfigurationsmöglichkeiten der Initialisierungsparameterdatei im OEM.
Die Instanz startet. Es werden die Werte aus dem PFILE gelesen. Es sind daher keine dynamischen Änderungen möglich.

Zusatz: Versuchen Sie ein PFILE aus dem SPFILE mittels SQL-Anweisung zu erstellen. Versuchen Sie ein SPFILE aus dem erstellten PFILE zu erstellen. Testen Sie beide Möglichkeiten (OEM und SQL-Anweisung).

Erstellen Sie ein PFILE manuell und binden das SPFILE darin ein.

1. Fahren Sie die Instanz herunter.
2. Erstellen Sie im Verzeichnis `/pfile` eine Textdatei, die Sie `init_SID.ora` (z.B. `init_testdb29.ora`) nennen.
3. Schreiben Sie in diese Datei nur folgende Zeile:
   `spfile= /spfile/spfile_SID.ora` (z.B. `spfile_testdb29.ora`)
4. Fahren Sie die Instanz mit diesem PFILE hoch:
   `startup pfile= /pfile/init_testdb29.ora`

Was passiert und warum?
Die Instanz fährt mit dem eingebundenen PFILE hoch.

A.2.2 Hoch- / Herunterfahren

1. Legen Sie einen neuen Benutzer `test` mit dem Kennwort `test` an.

2. Geben Sie ihm die Rechte `connect` und `select any table`.

3. Melden Sie sich über das SQL-Worksheet als Benutzer `test` an.

4. Melden Sie sich weiterhin über das SQL-Worksheet als Benutzer `sys` an.

5. Fahren Sie `sys` als Ihre Instanz normal herunter (`shutdown normal;`). Was passiert und warum?
   Es ist kein `shutdown normal` möglich, das der User `test` noch angemeldet ist.

6. Wechseln Sie in das Worksheet des Benutzers `test`.

7. Melden Sie den Benutzer `test` ab (`disconnect`).

8. Wechseln Sie in das Worksheet des Benutzers `sys`.
   Was passiert und warum?
   Die Instanz fährt herunter.

Spielen Sie das Szenario mit den Optionen `shutdown immediate` und `shutdown abort` durch. Was passiert und warum?
Bei `immediate` und `abort` fährt die Instanz trotz `connection` vom User `test` herunter.

Starten Sie eine heruntergefahrene Instanz mittels der verschiedenen Start-Optionen. Wechseln Sie von der niedrigsten Stufe in die jeweils höhere (`NOMOUNT`, `MOUNT`, `OPEN`). Bei welchen Optionen kann sich der Benutzer `test` wieder einloggen?
Das Einloggen klappt erst bei `OPEN`.

Nun werden Sie die Datenbank im eingeschränkten Modus hochfahren.

1. Starten Sie Ihre Instanz mit `startup restrict;`

2. Versuchen Sie sich als User `test` anzumelden. Was passiert und warum?
   Der User `test` kann sich nicht anmelden (Restricted Session).

3. Ändern Sie Ihre Instanz vom eingeschränkten in den nicht eingeschränkten Modus.
   `alter system disable restricted session;`

4. Versuchen Sie sich erneut als User `test` anzumelden. Was passiert und warum?
   Der User `test` kann sich anmelden.

5. Mit welchem Befehl ändern Sie eine aktuelle Sitzung von normalen Modus in den eingeschränkten Modus?
   `alter system enable restricted session;`

A.2.3 Control Files spiegeln

In dieser Übung Sie die Kontroldateien Ihrer Datenbank spiegeln, um eine höhere Ausfallsicherheit zu gewährleisten. Die Kontrolldateien sollten dabei auf unterschiedliche Datenträger gespeichert werden.

1. Ermitteln Sie aus der Data Dictionary View `V$CONTROLFILE` den Speicherort der Kontroldateien. Notieren Sie sich die angegebenen Pfade.
2. Erstellen Sie drei neue Verzeichnisse. Bennennen Sie diese in /hdd1, /hdd2 und /hdd3. Diese Bezeichnungen sollen unterschiedliche Datenträger darstellen. Wenn es Ihnen möglich ist, sollten Sie diese Verzeichnisse auch als Mount-Points für unterschiedliche Datenträger verwenden.
3. Melden Sie sich mit dem SQL*Plus-Worksheet als `sys` an.
4. Ändern Sie Ihr SPFILE so, dass die neuen Pfadangaben für die Kontrolldateien darin angegeben werden. Bedenken Sie, dass die Pfadangaben für die Kontrolldateien keine dynamischen Werte des SPFILE sind.

```
alter systemset control_files =
 (
   '/hdd1/control01.ctl',
   '/hdd2/control02.ctl',
   '/hdd3/control03.ctl'
 )
 SCOPE=SPFILE
;
```

5. Fahren Sie Ihre Instanz herunter und verschieben (nicht kopieren) Sie die Kontrolldateien mittels Betriebssystem-Befehlen an die entsprechenden Verzeichnisse.
6. Starten Sie die Instanz wieder (`OPEN`). Ermitteln Sie aus der Data Dictionary View `V$CONTROLFILE` wieder den Speicherort der Kontroldateien. Wo befinden sich die drei Kontroldateien jetzt?
   /hdd1/control01.ctl,/hdd2/control02.ctl,/hdd3/control03.ctl

Zusatz:
Fahren Sie die Instanz herunter. Benennen Sie die Kontrolldateien um und simulieren so den Ausfall sämtlicher Kontrolldateien. Läßt sich die Instanz starten?
Nein.

Simulieren Sie den Ausfall nur einer Kontrolldatei. Was passiert beim Start?
Es wird ein Lesefehler der Kontrolldatein angezeigt (`alert.log`). Es wird aber versucht die DB zu mounten. Dies gelingt aber nicht.

A.2.4 Redo Log Dateien

In dieser Übung werden Sie neue Redo-Log-Member zu bereits bestehenden Redo-Log-Gruppen hinzufügen (spiegeln). Das Spiegeln der Member dient zur Erhöhung der Ausfallsicherheit und der Verfügbarkeit. Weiterhin werden Sie Redo-Log-Gruppen hinzufügen. Dies kann die Performance steigern.

1. Ermitteln Sie die Datein Ihrer Log-Files aus den Data Dictionary Views `V$LOG` und `V$LOGFILE`. Wieviele Gruppen und Member besitzt Ihre Datenbank?
   Es sind drei Gruppen mit jeweils einem Member.

2. Erstellen Sie zwei neue Verzeichnisse. Benennen Sie die Ordner `/redoSP` und `/redo4`.

Teil 1
Fügen Sie den bestehenden Redo-Log-Gruppen neue Member hinzu und spiegeln Sie diese. Die Member sollten sich auf unterschiedlichen Datenträgern befinden.

1. Melden Sie sich mittels SQL*Plus Worksheet als `sys` an.

2. Fügen Sie zu jeder Redo-Log-Gruppe einen weiteren Member hinzu. Legen Sie die Member so an, dass die neuen Member im Ordner `/redoSP` gespeichert werden. Benennen Sie den neuen Redo-Log-Member der ersten Gruppe mit `redo01a.log`, den der zweiten Gruppe mit `redo02a.log` und den der dritten Gruppe mit `redo03a.log`.

```
alter database
  add logfile member
    '/redoSP/redo01a.log' to group 1,
    '/redoSP/redo02a.log' to group 2,
    '/redoSP/redo03a.log' to group 3
;
select group#,member,status from v$logfile;
```

3. Prüfen Sie Ihr Ergebnis mit Hilfe der View V$LOGFILE.
   Was stellen Sie fest?
   Alle sind als INVALID gekennzeichnet.

Teil 2
Nun werden Sie eine neue Redo-Log-Gruppe hinzufügen.

1. Melden Sie sich mittels SQL*Plus Worksheet an Ihrer Datenbank als `sys` an.

2. Fügen Sie die vierte Gruppe hinzu. Diese soll ebenfalls zwei Member mit einer Größe von 50MB besitzen. Speichern Sie den ersten Member im Ordner `/redo4` und den zweiten in `/redoSP` ab. Bennen Sie die Redo-Log-Dateien mit `redo04a.log` (erster Member) bzw. `redo04b.log` (zweiter Member).

```
alter database
  add logfile group 4
    ('/redo4/redo04a.LOG',
     '/redoSP/redo04b.LOG')
    size 50M
;
```

3. Prüfen Sie Ihr Ergebnis mit Hilfe der Views V$LOGFILE und V$LOG.

```
select group#,member,status from v$logfile;
select group#,status from v$log;
```

Was stellen Sie fest?
Alle sind da.

Teil 3
Nun werden Sie die Größe der Redo-Log-Gruppen 1 bis 3 so ändern, dass Ihre Größe nur noch 50MB beträgt.

1. Melden Sie sich mittels SQL*Plus Worksheet als `sys` an.

2. Prüfen Sie den Status Ihrer Redo-Log-Gruppe, die Sie löschen möchten. Beachten Sie, dass aktive Redo-Log-Dateien nicht gelöscht werden können. Switchen Sie, wenn nötig zur nächsten Redo-Log-Gruppe.

```
select group#,status from v$log;
```

3. Löschen Sie Ihre erste Redo-Log-Gruppe.

```
alter database drop logfile group 1;
```

4. Legen Sie eine neue Redo-Log-Gruppe 1 mit der Größe von 50MB an.

```
alter database
  add logfile group 1
    ('/oracle/oradata/testdb29/redo01a.LOG',
     '/redoSP/redo01b.LOG')
    size 50M;
```

Was passiert und warum?
Das Alert-Log zeigt die Änderung an, die Views `V$LOGFILE` und `V$LOG` dagegen nicht.

5. Fügen Sie Ihrem Erstellungsbefehl das abschließende Wort `REUSE` hinzu.

```
alter database
  add logfile group 1
  ('/oracle/oradata/testdb29/redo01a.LOG',
   '/redoSP/redo01b.LOG')
  size 50M reuse;
```

6. Nach kurzer Zeit erscheint die Meldung: Datenbank geändert. Prüfen Sie, ob die Redo-Log-Gruppe wirklich in der richtigen Größe an den richtigen Orten erstellt wurde.

```
select group#,status from v$log;
select group#,member,status from v$logfile;
```

7. Verfahren Sie mit den anderen Gruppen ebenso.

Zusatz
Prüfen Sie, ob es möglich ist Redo-Log-Gruppen mit einer unterschiedlichen Anzahl von Membern anzulegen. Welcher Sinn könnte dahinter stecken?

A.2.5 Index

Im Folgenden werden Sie die Funktionsweise von Indizes und deren Verwendbarkeit kennen lernen. Arbeiten Sie mit dem SQL*Plus Woksheet.

Melden Sie sich an Ihrer Datenbank als sys an.

Erstellten Sie eine Tabelle index_test im lokal verwalteten Tablespace ts_index_test mit einer Datendatei von 500 MB. Die Tabelle soll eine Spalte name besitzen mit dem Datentyp char(2000).

```
create tablespace ts_index_test
  datafile '/var/tablespaces/ts_index_test.dbf'
  size 500M extent management local;
create table index_test (name char(2000)) tablespace ts_index_test;
```

Geben Sie folgendes Kommando ein:

```
set timing on;
```

Was bewirkt dieses Kommando?

```
set timing on;
select * from index_test;
...
Abgelaufen: 00:00:00.02
```

Geben sie folgende Namen ein: Klaus, Silke, Manu, Stefan, Alex.

```
insert into index_test values ('Klaus');
insert into index_test values ('Silke');
insert into index_test values ('Manu');
insert into index_test values ('Stefan');
insert into index_test values ('Alex');
```

Fügen Sie weitere Datensätze der Tabelle hinzu, indem Sie folgenden Befehl 12 mal verwenden:

```
-- 12x
insert into index_test select * from index_test;
--
select count(*) from index_test;
```

Danach haben Sie 20480 Datensätze in Ihrer Tabelle. Fügen Sie nun einen Datensatz mit Anna als Namen ein.

```
insert into index_test values ('Anna');
commit;
```

Sie werden nun eine Abfrage starten, bei der ein Full Table Scan durchgeführt werden muss. Dabei wird die gesamte Tabelle nach den entsprechenden Datensatz durchsucht. Wie lange benötigt die Abfrage zur Ausführung?

```
select * from index_test where name = 'Anna';
...
Abgelaufen: 00:09:35.00
```

Erstellen Sie nun einen normalen Index auf die Spalte name Ihrer Tabelle test.

```
create index id1 on index_test(name);
```

Wiederholen Sie die in Punkt 7 getätigte Abfrage und notieren Sie sich die zur Ausführung benötigte Zeit. Vergleichen Sie beide Zeiten miteinander und interpretieren Sie das Ergebnis.

```
select * from index_test where name = 'Anna';
...
Abgelaufen: 00:00:15.08
```

Löschen Sie den Index, die Tabelle und den Tablespace.

```
drop index id1;
drop table index_test;
drop tablespace ts_index_test;
```

A.2.6 Constraints

Im Folgenden werden Sie die Funktionsweise des Deaktivierens und der verschiedenen Aktivierungszustände von Constraints kennen lernen. Arbeiten Sie folgende Übung mit dem SQL*Plus Woksheet durch.

Melden Sie sich als `sys` an.

Erstellten Sie eine Tabelle constr_test im lokal verwalteten Tablespace ts_constr_test mit einer Datendatei von 50 MB. Die Tabelle soll eine Spalte name besitzen mit dem Datentyp varchar(20).

```
create tablespace ts_constr_test
  datafile '/var/tablespaces/ts_constr_test.dbf'
  size 50M extent
  management local
;
create table constr_test (name varchar(20))
  tablespace ts_constr_test
;
```

Geben sie folgende Namen ein: Klaus, Silke, Manu, Stefan, Alex, Ines.

```
insert into constr_test values ('Klaus');
insert into constr_test values ('Silke');
insert into constr_test values ('Manu');
insert into constr_test values ('Stefan');
insert into constr_test values ('Alex');
insert into constr_test values ('Ines');
```

Erstellen Sie ein Check-Constraint für die Spalte name. Es soll verhindern werden, dass der Name Anna und Berta eingefügt werden kann.

```
alter table constr_test
  add constraint ck_name_constr_test
  check (name not in ('Anna', 'Berta'))
;
```

Versuchen Sie nun den Namen Anna in die Tabelle einzufügen.
Was passiert und warum?

```
insert into constr_test values ('Anna');
FEHLER in Zeile 1:
ORA-02290: Verstoß gegen CHECK-Regel (SYS.CK_NAME_TEST1)
```

Schalten Sie nun die Constraint-Prüfung aus.

```
alter table constr_test disable validate constraint ck_name_constr_test;
```

Versuchen Sie nun wiederum den Namen Anna in Ihre Tabelle einzufügen.
Was passiert und warum?

```
insert into constr_test values ('Anna');
FEHLER in Zeile 1:
ORA-25128: Kein Einfügen/Aktualisieren/Löschen bei Tabelle mit
deaktiviertem und validiertem Constraint (SYS.CK_NAME_TEST1)
```

Schalten Sie die Constraint-Prüfung nun wieder ein. Aktivieren Sie sie so, dass bestehende Datensätze nicht durch das Constraint geprüft werden. Was passiert und warum?

```
alter table constr_test enable novalidate constraint ck_name_constr_test;
insert into constr_test values ('Anna');
FEHLER in Zeile 1:
ORA-02290: Verstoß gegen CHECK-Regel (SYS.CK_NAME_TEST1)
```

Können Sie den Namen Berta der Tabelle hinzufügen?

```
insert into constr_test values ('Berta');
FEHLER in Zeile 1:
ORA-02290: Verstoß gegen CHECK-Regel (SYS.CK_NAME_TEST1)
```

Ändern Sie den Zustand des Constraints so ab, dass bestehende Datensätze auf Übereinstimmung geprüft werden. Was passiert und warum?

```
alter table constr_test enable validate constraint ck_name_constr_test;
Tabelle wurde geändert.
```

A.2.7 Benutzerverwaltung

Sie werden drei Benutzer namens anna, berta und carla erstellen.

1. Erstellen Sie den User anna. Das Passwort ist auch anna.

```
create user anna identified by anna;
grant connect, create session to anna;
```

2. Melden Sie sich unter SQLPLUS als anna@testdb29 an. Was passiert?

```
sqlplus anna@testdb29
Kennwort eingeben: anna
SQL>
```

3. Versuchen Sie eine Tabelle zu erstellen.

```
SQL> create table annatest(s1 int);
FEHLER in Zeile 1:
ORA-01031: Unzureichende Berechtigungen
```

4. Melden Sie sich als User anna wieder ab.

```
SQL> exit
```

5. Geben Sie dem User anna unbegrenzten Speicherplatz auf dem Tablespace users.

```
alter user anna quota unlimited on users;
alter user anna default tablespace users
      quota unlimited on users;
```

6. Versuchen Sie wieder eine Tabelle zu erstellen. Was passiert und warum?

```
sqlplus anna@testdb29
Kennwort eingeben: anna
SQL> create table annatest(s1 int);
```

7. Fügen Sie zwei Datensätze in die Tabelle ein.

```
SQL> insert into annatest values(1);
SQL> insert into annatest values(2);
SQL> select * from annatest;
        S1
----------
         1
         2
```

8. Schließen Sie die Transaktion ab.

```
SQL> commit;
```

9. Erstellen Sie einen User berta und als berta eine Tabelle bertatest. Versuchen Sie als User berta die Tabelle annatest abzufragen.

```
create user berta identified by berta;
grant connect, create session to berta;
alter user berta quota unlimited on users;
alter user berta default tablespace users quota unlimited on users;
```

Was passiert und warum?

```
sqlplus berta@testdb29
Kennwort eingeben:
SQL> create table bertatest(s1 int);
SQL> select * from anna.annatest;
select * from anna.annatest
FEHLER in Zeile 1:
ORA-00942: Tabelle oder View nicht vorhanden
```

10. Versuchen Sie eine Lösung zu finden.

```
sqlplus anna@testdb29
SQL> grant select on annatest to berta;
Benutzerzugriff (Grant) wurde erteilt.

sqlplus berta@testdb29
SQL> select * from anna.annatest;
        S1
----------
         1
         2
```

11. Überprüfen Sie, ob es möglich ist im Schema eines anderesn Users eine Tabelle zu erstellen.

```
sqlplus berta@testdb29
SQL> create table anna.bertatest(s1 int);
FEHLER in Zeile 1:
ORA-01031: Unzureichende Berechtigungen
```

Kann nur durch SYS dem User anna zugewiesen werden (grant ... with admin option). sys darf natürlich Rechte für Objekte vergeben. anna kann nur für ihre Objekte die Berechtigungen weitergeben.

```
sqlplus anna@testdb29
SQL> grant select, insert, update, delete on anna.annatest to berta;
Benutzerzugriff (Grant) wurde erteilt.
```

A.2.8 Namensauflösung

Teil 1 – Local Naming

1. Öffnen Sie die Datei `sqlnet.ora`.
   Wie ist die Reihenfolge der Namensauflösung?
   TNSNAMES, ONAMES, HOSTNAME

2. Nennen Sie die Datei `tnsnames.ora` in `tnsnames.old` um. Versuchen Sie sich an dern Datenbank `testdb29` anzumelden. Was passiert und warum?
   ORA-12154 TNS: Der Servicename konnte nicht aufgelöst werden.

3. Nennen Sie die Datei `tnsnames.old` wieder in `tnsnames.ora` um. Was passiert und warum?
   Der Connect wird durchgeführt.

4. Öffnen Sie die Datei `sqlnet.ora`. Ändern Sie dort die folgende Zeile:
   `names.directory_path = (ONAMES, HOSTNAME)`
   Speichern Sie die Datei `sqlnet.ora`.

5. Versuchen Sie sich an der Datenbank `testdb29` anzumelden. Was passiert?
   ORA-12154 TNS: Der Servicename konnte nicht aufgelöst werden.

6. Öffnen Sie wieder die Datei `sqlnet.ora`. Ändern Sie dort die folgende Zeile:
   `names.directory_path = (TNSNAMES, ONAMES, HOSTNAME)`
   Speichern Sie die Datei `sqlnet.ora`.

7. Versuchen Sie sich wieder an der Datenbank `testdb29` anzumelden. Was passiert?
   Der Connect wird durchgeführt.

8. Öffnen Sie die Datei `tnsnames.ora`. Ändern Sie dort den Port für die Datenbank `testdb29` auf 1539. Versuchen Sie sich wieder an der Datenbank `testdb29` anzumelden. Was passiert?
   ORA-12541 TNS: Kein Listener.

9. Öffnen Sie die Datei `listener.ora`. Ändern Sie auch dort den Port für die Datenbank `testdb29` auf 1539. Starten Sie den Listener neu. Was passiert?
   Der Connect wird durchgeführt.

10. Ändern Sie die Port-Einstellungen in `listener.ora` und `tnsnames.ora` wieder zurück auf 1521

11. Erweitern Sie die Datei `tnsnames.ora`, so dass Sie sich auf die Datenbank Ihres Nachbarn verbinden können.

Teil 2 – Host Naming

1. Öffnen Sie die Datei `sqlnet.ora`. Ändern Sie dort die folgende Zeile:
   `names.directory_path = (HOSTNAME)`
   Speichern Sie die Datei `sqlnet.ora`.

2. Versuchen Sie sich an der Datenbank `testdb29` anzumelden. Was passiert?
   ORA-12154 TNS: Der Servicename konnte nicht aufgelöst werden.

3. Öffnen Sie die Datei `listener.ora`. Ändern Sie die Datei folgendermassen:

   ```
   ...
   (SID_DESC=
      (GLOBAL_DBNAME=pinguin)
      (ORACLE_HOME=/opt/oracle/product/9ir2)
      (SID_NAME=testdb29)
   )
   ...
   ```

   Starten Sie den Listener neu.

4. Verbinden Sie sich mit sqlplus mit der Datenbank `testdb29`.

   ```
   sqlplus /nolog
   connect sys/sys@pinguin
   ```

   Was passiert?
   Der Connect wird durchgeführt.

5. Warten Sie bis Ihr Nachbar soweit ist und versuchen Sie sich mit der Datenbank auf dessen Computer zu verbinden.

6. Stellen Sie die ursprüngliche Datei `listener.ora` wieder her.

7. Öffnen Sie die Datei `sqlnet.ora`. Brinden Sie dort die folgende Zeile wieder auf den alten Zustand:
   `names.directory_path = (TNSNAMES, ONAMES, HOSTNAME)`
   Speichern Sie die Datei `sqlnet.ora`.

A.2.9 Cold Backup im NOARCHIVELOG-Modus

In dieser Übung wollen wir folgendes Szenario durchspielen. Jeden Tag gegen 18:00 Uhr wir die Datenbank testdb29 mit Hilde eines Cold-Backups gesichert. Am Dienstag gegen 16:00 Uhr bemerken Sie, dass die Datei users01.dbf beschädigt ist. Sie müssen die Funktionsfähigkeit der Datenbank schnellstmöglich wieder herstellen.

Montag 18:00 Uhr – Cold Backup
Führen Sie ein Cold-Backup durch. Die Datenbank testdb29 soll im Modus NOARCHIVE-Log sein.

```
shutdown immediate;
startup mount;
alter database noarchivelog;
archive log stop;
```

Fahren Sie nun die Instanz herunter un kopieren Sie alle notwendigen Dateien in das Verzeichnis /backup. Fahren Sie anschließend die Instanz wieder hoch.

```
shutdown immediate;
host;
cp /oracle/oradata/testdb29/* /backup
exit
startup;
```

Dienstag 08:00 bis 15:59 – Arbeitslast
Simulieren Sie die Arbeitslast. Sie werden hierfür eine Tabelle anlegen, dort fünf Datensätze einfügen und drei Log-Switches ausführen.

```
create table unterricht (fachnummer number, fachname varchar2(20));
insert into unterricht values(1,'Deutsch');
insert into unterricht values(2,'Deutsch');
insert into unterricht values(3,'Englisch');
insert into unterricht values(4,'Mathe');
insert into unterricht values(5,'Zeichnen');
commit;
alter system switch logfile;
alter system switch logfile;
alter system switch logfile;
```

Dienstag 16:00 – Verlust der Datei users01.dbf
Simulieren Sie den Verlust der Datendatei users01.dbf.

```
shutdown immediate;
host;
rm /oracle/oradata/testdb29/users01.dbf
exit
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01157: Datendatei 3 kann nicht identifiziert/gesperrt werden
           Siehe DBWR-Trace-Datei.
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Wiederherstellung der Datenbank testdb29 (1. Versuch)
Als erstes versuchen Sie die Datei users01.dbf wiederherzustellen.

```
shutdown immediate;
host;
cp /backup/users01.dbf /oracle/oradata/testdb29
exit
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01113: Für Datei '3' ist eine Datenträger-Recovery notwendig
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Versuchen Sie nun, die seit der letzten Sicherung durchgeführten Transaktionen aus den Online-Redo-Log-Dateien wiederherzustellen. Es kann sein, dass bei Ihnen die Datei users01.dbf nicht die Dateindatei 3 ist.

```
recover datafile 3;
```

Warum schlägt dieses Vorhaben fehl?
Es müssen alle Datendateien wiederhergestellt werden.

Wiederherstellung der Datenbank testdb29 (2. Versuch)
Versuchen Sie alle Datendateien wiederherzustellen.

```
shutdown immediate;
host;
cp /backup/* /oracle/oradata/testdb29
exit
startup;
ORACLE-Instanz hochgefahren.
...
ORA-00214: Steuerdatei '/hd2/control02.ctl' Version 684
 nicht konsistent mit Datei
 '/oracle/oradata/testdb29/control01.ctl Version 657
```

Natürlich müssen noch die Multiplex-Control-Dateien an den entsprechenden Stellen und unter den entsprechenden Namen wiederhergestellt werden. Da die Control-Dateien immer identisch sind, kann die gesicherte Control-Datei aus dem Verzeichnis /backup an die anderen Speicherorte ohne weiteres kopiert werden.

```
shutdown immediate;
host;
cp /backup/control01.ctl /hd2/control02.ctl
cp /backup/control01.ctl /hd3/control03.ctl
exit
startup;
ORACLE-Instanz hochgefahren.
...
Datenbank geöffnet.
```

Nun testen Sie, ob die Tabelle unterricht noch vorhanden ist.

```
select * from unterricht;
              *
FEHLER in Zeile 1:
ORA-00942: Tabelle oder View nicht vorhanden.
```

A.2.10 Cold Backup im ARCHIVELOG-Modus

In dieser Übung wollen wir das Szenario der vorherigen Übung durchspielen. Die Datenbank wird aber in den Archive-Log-Modus versetzt.

```
select log_mode from v$database;
LOG_MODE
------------
NOARCHIVELOG
1 Zeile wurde ausgewählt.

shutdown immediate;

sqlplus /nolog
connect sys/sys as sysdba

startup mount;
alter database archivelog;
```

Manuell:

```
alter system archive log start;
alter system set log_archive_start=true scope=spfile;
alter database open;
```

Montag 18:00 Uhr – Cold Backup
Im folgenden werden Sie ein Cold-Backup durchführen. Fahren Sie nun die Instanz herunter und kopieren Sie alle notwendigen Dateien in das Verzeichnis /backup. Fahren Sie anschließend die Instanz wieder hoch.

```
shutdown immediate;
host;
cp /oracle/oradata/testdb29/* /backup
exit
startup;
```

Dienstag 08:00 bis 15:59 – Arbeitslast
Im folgenden werden Sie die Arbeitslast vom Dienstag simulieren. Legen Sie hierfür eine Tabelle an, fügen dort fünf Datensätze ein und führen drei Log-Switches aus.

```
create table unterricht (fachnummer number, fachname varchar2(20));
insert into unterricht values(1,'Deutsch');
insert into unterricht values(2,'Deutsch');
insert into unterricht values(3,'Englisch');
insert into unterricht values(4,'Mathe');
insert into unterricht values(5,'Zeichnen');
commit;
alter system switch logfile;
alter system switch logfile;
alter system switch logfile;
```

Dienstag 16:00 – Verlust der Datei users01.dbf
Simulieren Sie den Verlust der Datendatei users01.dbf.

```
shutdown immediate;
host;
rm /oracle/oradata/testdb29/users01.dbf
exit
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01157: Datendatei 3 kann nicht identifiziert/gesperrt werden
           Siehe DBWR-Trace-Datei.
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Wiederherstellung der Datenbank testdb29
Versuchen Sie die Datei users01.dbf wiederherzustellen.

```
shutdown immediate;
host;
cp /backup/users01.dbf /oracle/oradata/testdb29
exit
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01113: Für Datei '3' ist eine Datenträger-Recovery notwendig
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Als nächstes versuchen Sie nun, die seit der letzten Sicherung durchgeführten Transaktionen aus den Online-Redo-Log-Dateien wiederherzustellen. Es kann sein, dass bei Ihnen die Datei users01.dbf nicht die Dateindatei 3 ist.

```
recover datafile 3;
```

Warum sfunktioniert dieses Vorhaben nunmehr?

A.2.11 Control File skripten / wiederherstellen

In dieser Übung werden wir untersuchen, wie sich die Datenbank verhält, wenn alle Control-Files fehlen. Voraussetzung hierbei ist, dass alle Data Files noch vorhanden sind. Glücklicherweise haben wir vorher ein Skript erstellt, welches die Control Files neu erstellen kann. Ansonsten wäre alles etwas aufwendiger.

1. Skripten Sie das Control File.

   ```
   alter database backup controlfile to trace;
   ```

2. Schauen Sie in den Ordner `/opt/oracle/admin/testdb29/udump` nach einer Datei, die eben erstellt wurde.
3. Öffnen Sie diese Datei in einem Editor. Sie sehen die CREATE CONTROLFILE-Anweisung, die Oracle automatisch erstellt hat.
4. Fahren Sie die Instanz herunter und löschen Sie alle Control Files.
5. Starten Sie die Instanz wieder. Was passiert und warum?
6. Kopieren Sie nun die komplette CREATE CONTROLFILE-Anweisung aus dem Trace-File in das SQL-Plus-Fenster und führen Sie diese Anweisung aus.
7. Versuchen Sie nun die Datanbank zu öffnen.

   ```
   alter database open;
   ```

   Was passiert?
   Alles ist wieder gut.

A.2.12 Vollsicherung mit RMAN

Sie werden mit RMAN eine inkonsistente Vollsicherung der Datenbank testdb29 durchführen. Stellen Sie sicher, dass das Verzeichnis /backup existiert. Starten Sie RMAN.

```
RMAN target sys/sys@testdb29 nocatalog
```

Geben Sie folgende Anweisung ein:

```
run {
   allocate channel s1 type disk format '/backup/b_%u_%s_%p';
   backup database;
}
```

Erstellen Sie die Tabelle abteilung und tragen dort zwei Abteilungen ein. Führen Sie drei Log-Switches durch. Die Änderungen befinden sich demnach nicht mehr in den Online-Redo-Log-Dateien.

```
create table abteilung (abtnr int, abtname varchar2(30));
insert into abteilung values (1,'Forschung');
insert into abteilung values (2,'Entwicklung');
commit;
alter system switch logfile;
alter system switch logfile;
alter system switch logfile;
```

Simulieren Sie den Verlust der Datendatei users01.dbf.

```
shutdown immediate;
host;
rm /oracle/oradata/testdb29/users01.dbf
exit
startup;
...
Datenbank mit Mount angeschlossen.
ORA-01157: Datendatei 3 kann nicht identifiziert/gesperrt werden
           Siehe DBWR-Trace-Datei.
ORA-01110: Datendatei 3: '/oracle/oradata/testdb29/users01.dbf'
```

Im welchen Zustand befindet sich die Instanz?
MOUNT

Stellen Sie die Datei users01.dbf mit RMAN wieder her.

```
RMAN target sys/sys@testdb29 nocatalog
```

Geben Sie folgende Anweisung ein:

```
run {
  allocate channel s1 type disk format '/backup/b_%u_%s_%p';
  restore datafile '/oracle/oradata/testdb29/users01.dbf';
  recover datafile '/oracle/oradata/testdb29/users01.dbf';
}
```

Wechseln Sie zu SQLPLUS und öffnen Sie die Datenbank.

```
alter database open;
```

A.3 Tuning – Übungen und Lösungen

A.3.1 Tools für die Leistungsüberwachung

In der folgenden Übung werden Sie Hilfsmittel und Tools unter Oracle kennen lernen, mit denen man wichtige Statistiken und Auswertungen einer Datenbank durchführen kann.

Grundlage sind die Scripte zur Erstellung der Nordwind-DB.

1. Melden Sie sich als sys and der Datenbank an.

2. Erstellen Sie die Tabelle employees aus dem Skript `employees.txt`. Wie viel Datensätze hat die Tabelle employees?

```
start ~/1ZO-033/Northwind-Skript/employees.txt

select count(*) from employees;

  COUNT(*)
----------
         9
```

3. Führen Sie folgende Anweisung aus, um die Ausgabe des Ausführungsplanes für die jeweiligen Anweisungen durchzusetzen.

```
set autotrace on explain;
```

4. Was passiert und was ist die Ursache?

```
SP2-0613: Format oder Existenz von PLAN_TABLE kann nicht überprüft werden
SP2-0611: Fehler beim Initialisieren von EXPLAIN - Bericht
```

5. Erstellen Sie nun die Plan-Tabelle, indem Sie das Skript `utlxplan.sql` (den Speicherort der Datei müssen Sie selber herausfinden) in Ihrem aktuellen Fenster ausführen. Führen Sie anschließend erneut die Anweisung aus.

```
start /oracle/ora92/rdbms/admin/utlxplan.sql

set autotrace on explain;
```

6. Was passiert? Wo lag die Datei utlxplan.sql?

```
/oracleora92/rdbms/admin/utlxplan.sql
```

7. Führen Sie eine Abfrage aus, welche alle Datensätze und alle Attribute (Spalten) der Tabelle employees zurückgibt. Was sagt Ihnen der Ausführungsplan?

```
select * from employees;

Ausführungsplan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'EMPLOYEES'
```

8. Nun wollen wir uns auch jeweils die Zeit anzeigen lassen, die eine Abfrage benötigte. Führen Sie hierfür folgende Anweisung aus:

```
set timing on;
```

9. Führen Sie eine Abfrage aus, welche alle Datensätze und alle Attribute (Spalten) der Tabelle employees zurückgibt. Wie lange dauerte die Abfrage?

```
select * from employees;

Abgelaufen: 00:00:00.01

Ausführungsplan
----------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   TABLE ACCESS (FULL) OF 'EMPLOYEES'
```

10. Erzeugen Sie eine Tabelle namens empl_cop, welche genau die Spalten und die Datensätze der Employees enthält. Wie lautet die Anweisung hierfür?

```
create table empl_cop as select * from employees;
```

11. Verdoppeln Sie die Anzahl der Datensätze in dieser Tabelle empl_cop insgesamt 16 mal. Wie viel Datensätze haben Sie nunmehr?

```
-- 16 mal:
insert into empl_cop select * from empl_cop;
commit;

select count(*) from empl_cop;
  COUNT(*)
----------
    589824
```

12. WICHTIG: Alles weitere bezieht sich jeweils immer auf die empl_cop!
Schreiben Sie eine Abfrage, welche alle Employees und daneben die Anzahl der Datensätze der Employees ausgibt. Wie lautet die Abfrage und wie lange dauert diese? Überlegen Sie, warum diese Abfrage so aufwendig ist. Wir werden später eine ähnliche Abfrage so optimieren, dass die Ausführung unter 1/10 Sekunde dauert.

Beispielergebnismenge

```
LASTNAME               COUNT(*)
-------------------- ----------
Buchanan                  65536
Callahan                  65536
Davolio                   65536
Dodsworth                 65536
Fuller                    65536
King                      65536
Leverling                 65536
Peacock                   65536
Suyama                    65536
```

```
select lastname, count(*) from empl_cop group by lastname;

Abgelaufen: 00:00:20.09

Ausführungsplan
----------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (GROUP BY)
2  1     TABLE ACCESS (FULL) OF 'EMPL_COP'
```

13. Löschen Sie die Tabelle employees und auch die Tabelle Empl_cop wieder.

```
drop table employees;
drop table empl_cop;
commit;
```

Im folgenden werden Sie das Tool tkprof etwas genauer kennen lernen. Führen Sie hierfür folgende Schritte durch:

1. Erstellen Sie die Tabelle customers aus dem Skript customers.txt. Wie viel Datensätze hat die Tabelle customers?

```
start ~/1ZO-033/Northwind-Skript/customers.txt
select count(*) from customers;

  COUNT(*)
----------
        91

Abgelaufen: 00:00:00.00

Ausführungsplan
----------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (AGGREGATE)
2  1     TABLE ACCESS (FULL) OF 'CUSTOMERS'
```

2. Nun muss das Tracing eingeschaltet werden, um eine Trace-Datei als Grundlage für tkprof zu haben. Führen Sie hierfür folgenden Befehl aus:

```
alter session set sql_trace = true;
```

3. Mit welchen Befehl hätten Sie diese Einstellung permanent machen können?

```
alter system set sql_trace=true scope=spfile;
```

4. Wo befindet sich der Ordner, in dem die Trace-Datei gespeichert wird?

```
/oracle/admin/pingudb/udump/pingudb_ora_2404.trc
```

5. Wie könnten Sie diesen Ort verändern, auf eine andere Platte zum Beispiel?

```
USER_DUMP_DEST
```

Andere Platte entsprechend Inhalt von USER_DUMP_DEST mounten. ;-)

6. Ermitteln Sie die Kunden, die aus Berlin kommen

```
select * from customers where city = 'Berlin';

1 Zeilen ausgewählt.

Abgelaufen: 00:00:00.00

Ausführungsplan
----------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   TABLE ACCESS (FULL) OF 'CUSTOMERS'
```

Ermitteln Sie die Kunden, die aus London kommen

```
select * from customers where city = 'London';

6 Zeilen ausgewählt.

Abgelaufen: 00:00:00.03

Ausführungsplan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'CUSTOMERS'
```

Ermitteln Sie die Kunden, die aus Paris kommen

```
2 Zeilen ausgewählt.

Abgelaufen: 00:00:00.01

Ausführungsplan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'CUSTOMERS'
```

8. Nun sollten Sie das Tracing ausschalten.

```
alter session set sql_trace = false;
```

9. Schauen Sie nun nach, ob die Trace-Datei erstellt wurde.

ja

10. Führen Sie tkprof an der Unix-Shell aus, um sich eine lesbare Datei erstellen zu lassen.

```
tkprof.exe /oracle/admin/pingudb/udump/pingudb_ora_2404.trc /tmp/1.txt
```

11. Öffnen Sie nun die eben erstellte Textdatei und schauen Sie sich die Ergebnisse an

Zusatzaufgaben 1

Sicher kommt es vor, dass eine Abfrage so lange dauert, dass es sinnvoller wäre, diese Session abzubrechen. Hierfür können Sie folgendes machen: Melden Sie sich als Scott an und führen Sie eine lange Abfrage aus (z. B. Cross Join über order_details)

Machen Sie eine weitere Session auf:

Führen Sie folgende Abfrage aus, um den Benutzer herauszufinden, dessen Session beendet werden soll select username,sid,serial from v$session;

```
USERNAME                          SID    SERIAL#
------------------------------ ---------- ----------
                                       1          1
                                       2          1
...
SYS                                    9         34
SCOTT                                 12        242
```

Beenden Sie die Session folgendermassen (SID, SERIAL):

```
alter system kill session '12,242';
```

A.3.2 Indizes

In der folgende Übung werden Sie Indizes erstellen, anpassen und natürlich auch wieder löschen. Sie werden beobachten, ob der Index benutzt wird oder nicht.

WICHTIG: Legen Sie sich ein Cold-Backup Ihrer Datenbank an, um im Fehlerfall die Datenbank schnell wiederherstellen und weiterarbeiten zu können.

Bitte führen Sie folgende Schritte als Vorbereitung für die Übung durch.
Falls Sie die Plan-Tabelle in einer der vorigen Übungen noch nicht erstellt haben, so tun Sie dies bitte jetzt (utlxplan.sql ggf. ausführen).
Aktivieren Sie die Anzeige der Zeit und des Ausführungsplanes. Geben Sie hierfür folgendes ein:

```
set autotrace on exp;
set timing on;
```

Führen Sie folgende kleine Anweisung aus um zu sehen, ob die Einstellungen ordnungsgemäß umgesetzt wurden:

```
select * from dual;
```

Als Ergebnis müsste eine Zeitangabe und ein Ausführungsplan erscheinen

Im folgenden wollen wir untersuchen, wie lange die Suche nach einem Datensatz einer Tabelle bei einem Full Table Scan dauert und anschließend diese Prozedur durch einen Index verbessern:

1. Erstellen Sie die Tabelle order_det_prod, welche aus der orderid, productname, productid, quantity und unitprice (aus products und order_details) besteht. Verdoppeln Sie die Anzahl der Datensätze jeweils 12 mal. Wie viel Datensätze haben Sie nun?

```
create table order_det_prod as
  select o.orderid,p.productname,p.productid,o.quantity,p.unitprice
    from products p,order_details o
    where p.productid = o.productid
;
-- 12 mal:
insert into order_det_prod select * from order_det_prod;
commit;

select count(*) from order_det_prod;
  COUNT(*)
----------
   8826880
Abgelaufen: 00:05:32.09

Ausführungsplan
----------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (AGGREGATE)
2  1     TABLE ACCESS (FULL) OF 'ORDER_DET_PROD'
```

2. Ermitteln Sie die größte Bestellnummer. Wie heißt die Abfrage und wie lange dauert die Abfrage? Mit welcher Methode wurde dieser Datensatz ermittelt?

```
select max(orderid) from order_det_prod;

MAX(ORDERID)
------------
       11077

Abgelaufen: 00:01:06.05

Ausführungsplan
----------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (AGGREGATE)
2  1     TABLE ACCESS (FULL) OF 'ORDER_DET_PROD'
```

3. Erstellen Sie einen Index namens id1 basierend auf der Spalte orderid.

```
create index id1 on order_det_prod(orderid);

Index wurde angelegt.

Abgelaufen: 00:17:14.03
```

4. Ermitteln Sie die größte Bestellnummer. Wie lange dauert die Abfrage nunmehr? Mit welcher Methode wurde dieser Datensatz ermittelt?

```
select max(orderid) from order_det_prod;

MAX(ORDERID)
------------
       11077

Abgelaufen: 00:00:00.00
```

```
Ausführungsplan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   SORT (AGGREGATE)
   2    1     INDEX (FULL SCAN (MIN/MAX)) OF 'ID1' (NON-UNIQUE)
```

5. Wie oft wurde bisher 'Chai' bestellt (bitte keine Funktion wie lower etc. benutzen)? Wie schnell wurde die Abfrage ausgeführt?

```
select count(*) from order_det_prod where productname='Chai';

  COUNT(*)
----------
    155648

Abgelaufen: 00:01:07.05

Ausführungsplan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   SORT (AGGREGATE)
   2    1     TABLE ACCESS (FULL) OF 'ORDER_DET_PROD'
```

6. Erstellen Sie einen geeigneten Index namens id2 und führen Sie die Abfrage erneut aus. Wie schnell wurde die Abfrage nunmehr ausgeführt?

```
create index id2 on order_det_prod(productname);

Index wurde angelegt.
Abgelaufen: 00:25:17.03
...
```

7. Erstellen Sie eine Tabelle namens emp_cop basierend auf der employees-Tabelle. Die Tabelle sollte identisch zur employees sein (gleiche Spalten, gleiche Zeilen)

```
create table emp_cop as select * from employees;
```

8. Erstellen Sie einen Index namens id3 auf Basis von lastname

```
create index id3 on emp_cop(lastname);
```

9. Nun werden Sie den Index id3 analysieren, dass heißt, den Prozentsatz von ungenutztem (durch Löschen) zu dem gesamten Platz ermitteln.

```
ANALYZE INDEX id3 VALIDATE STRUCTURE;
select (del_lf_rows_len/lf_rows_len)*100 from index_stats;

(DEL_LF_ROWS_LEN/LF_ROWS_LEN)*100
---------------------------------
                                0
```

0% gelöschte Rows.

10. Löschen Sie den ältesten Mitarbeiter. Wie viel Datensätze wurden gelöscht?

```
delete from emp_cop where birthdate =
  (
    select min(birthdate) from emp_cop
  )
;
1 Zeile wurde gelöscht.
```

11. Nun werden Sie den Index analysieren, dass heißt, den Prozentsatz von ungenutztem (durch Löschen) zu dem gesamten Platz ermitteln. Legen Sie Ihre Schlussfolgerungen dar.

```
ANALYZE INDEX id3 VALIDATE STRUCTURE;
select (del_lf_rows_len/lf_rows_len)*100 from index_stats;

(DEL_LF_ROWS_LEN/LF_ROWS_LEN)*100
---------------------------------
                       11,0465116
```

11% gelöschte ROWs

12. Nun werden Sie ein rebuild des Index durchführen. Führen Sie diesen Befehl bitte auch durch.

```
ALTER INDEX id3 REBUILD;
```

13. Nun werden Sie den Index analysieren, dass heißt, den Prozentsatz von ungenutztem (durch Löschen) zu dem gesamten Platz ermitteln. Legen Sie Ihre Schlussfolgerungen dar:

```
ANALYZE INDEX id3 VALIDATE STRUCTURE;
select (del_lf_rows_len/lf_rows_len)*100 from index_stats;

(DEL_LF_ROWS_LEN/LF_ROWS_LEN)*100
---------------------------------
                                 0
```

Im folgenden werden Sie einen **Bitmap-Index** erstellen und testen:

1. Erweitern Sie die Tabelle order_det_prod um eine Spalte namens done (varchar2(1)).

```
alter table order_det_prod add(done varchar2(1));
```

2. Für alle Orderids unter 11000 soll Done 'N' sein und für alle darüber soll Done 'Y' sein.

```
update order_det_prod set done = 'N' where orderid < 11000;
update order_det_prod set done = 'Y' where orderid > 11000;
commit;
```

Hier geht der Rechner in die Knie!

3. In wie vielen Datensätzen steht bei Done Y drin?

```
select count(*) from order_det_prod where done='Y';
  COUNT(*)
----------
    880640
```

4. Erstellen Sie einen Bitmap-Index auf der Spalte Done.

```
create bitmap index id4 on order_det_prod(done);
```

5. In wie vielen Datensätzen steht bei Done N drin (Bitte benutzen Sie kein * beim Zählen)? Schreiben Sie hierfür eine Abfrage und messen Sie die Zeit.

```
select count(done) from order_det_prod where done='N';
```

Im folgenden werden Sie einen **funktionsbasierten Index** kennen lernen.

1. Ermitteln Sie alle Datensätze, bei denen der Productname Chaiïst. Hierbei soll es egal sein, ob das Wort groß oder klein geschrieben wurde.

```
select count(*) from order_det_prod
  where lower(productname) = 'chai'
;
```

2. Erstellen Sie einen normalen Index, der die Suche beschleunigen soll. Was passiert und warum?

```
create index id5 on order_det_prod(productname);
```

3. Löschen Sie diesen Index und schreiben Sie einen funktionsbasierten Index. Was passiert und warum?

```
drop index id5;

create index ... lower(..).. ;
```

Zusatzaufgabe 1:

Schreiben Sie eine Abfrage, die Ihnen angibt, welche Indizes auf der Tabelle customers liegen und welche Spalten diese umfassen.

select mit Hint

Zusatzaufgabe 2:

Löschen Sie den Bitmap-Index auf Done und erstellen Sie anstelle dessen einen normalen Index. Führen Sie die entsprechenden Messungen erneut durch. Benutzen auch eventuell Hints.

Zusatzaufgabe 3:

Erstellen Sie eine große Tabelle (ca. 3 bis 4 Mio Datensätze) und erstellen Sie einen Index. Überprüfen Sie nun, ob ein Rebuild schneller geht als ein Drop und Recreate des Index. Überprüfen Sie des weiteren, ob während des Rebuilds weitergearbeitet werden kann. Überprüfen Sie außerdem, ob während des normalen Recreate (also Drop und Create) weitergearbeitet werden kann. Legen Sie Ihrer Ergebnisse kurz schriftlich dar.

A.3.3 Optimierer

Im folgenden werden Sie gespeicherte Ausführungspläne testen und einsetzen. Außerdem werden Sie den Optimierer und dessen Arbeitsweise kennen lernen. Führen Sie hierfür folgende Schritte durch:

1. Erstellen Sie die Tabelle order_details neu aus dem Script zur Nordwind-DB.

```
start ~/1ZO-033/Northwind-Skript/order_details.txt
```

2. Schalten Sie Autotrace ein.

```
set autotrace on explain;
```

3. Fügen Sie zur Tabelle order_details eine Spalte namens paketgroesse varchar(10) hinzu.

```
alter table order_details add(paketgroesse varchar(10));
```

4. Geben Sie für 1/3 der Datensätze den Wert 'klein' in die Spalte Paketgroesse.

```
update order_details set paketgroesse=mod(rownum,3);
update order_details set paketgroesse='klein' where paketgroesse='1';
```

5. Geben Sie für 1/3 der Datensätze den Wert 'mittel' in die Spalte Paketgroesse.

```
update order_details set paketgroesse='mittel' where paketgroesse='2';
```

6. Geben Sie für 1/3 der Datensätze den Wert 'gross' in die Spalte Paketgroesse.

```
update order_details set paketgroesse='gross' where paketgroesse='0';
commit;
select paketgroesse,count(*) from order_details group by paketgroesse;

PAKETGROES   COUNT(*)
---------- ----------
gross             718
klein             719
mittel            718
```

7. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'klein' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt? Begründen Sie Ihre Antwort.

```
select * from order_details where paketgroesse='klein';

719 Zeilen ausgewählt.

Ausführungsplan
----------------------------------------------------
0      SELECT STATEMENT Optimizer=CHOOSE
1    0   TABLE ACCESS (FULL) OF 'ORDER_DETAILS'
```

8. Erstellen Sie einen Index basierend auf der Spalte paketgroesse.

```
create index id1 on order_details(paketgroesse);
```

9. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'mittel' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt? Begründen Sie Ihre Antwort.

```
select * from order_details where paketgroesse='mittel';

718 Zeilen ausgewählt.

Ausführungsplan
----------------------------------------------------------
0      SELECT STATEMENT Optimizer=CHOOSE
1    0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS'
2    1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE)
```

Es wird RBO genommen, weil keine Analyse der Tabelle existiert.

10. Zwingen Sie den Optimierer, den cost based optimizer (CBO) bei der nun folgenden Abfrage zu nehmen. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'gross' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt? Begründen Sie Ihre Antwort.

```
select /*+ALL_ROWS*/ *
  from order_details
  where paketgroesse='gross'
;
Ausführungsplan
------------------------------------------------------------
0    SELECT STATEMENT Optimizer=HINT: ALL_ROWS (Cost=1 Card=6 Bytes=432)
1  0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS' (Cost=1 Card=6 Bytes=432)
2  1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE) (Cost=1 Card=2)
```

Es erfolgt eine Index-Scan weil CBO, Index vorhanden und es gibt keine Statistik.

11. Fügen Sie einen Datensatz hinzu, bei dem die Paketgröße 'riesig' ist.

```
insert into order_details values(11111,22,3,4,44,'riesig');
```

12. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'riesig' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt? Begründen Sie Ihre Antwort.

```
select * from order_details
  where paketgroesse='riesig'
;
Ausführungsplan
----------------------------------------------------------
0      SELECT STATEMENT Optimizer=CHOOSE
1    0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS'
2    1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE)
```

RBO, da noch keine Statistik vorhanden ist.

13. Zwingen Sie den Optimierer, den CBO bei der nun folgenden Abfrage zu nehmen. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'riesig' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt? Begründen Sie Ihre Antwort.

```
select /*+ALL_ROWS*/ *
  from order_details
  where paketgroesse='riesig'
;
Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=HINT: ALL_ROWS (Cost=1 Card=6 Bytes=432)
1  0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS' (Cost=1 Card=6 Bytes=432)
2  1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE) (Cost=1 Card=2)
```

Index-Scan, da keine Analyse,

14. Zwingen Sie den Optimierer, den rules based optimizer (RBO) bei der nun folgenden Abfrage zu nehmen. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'riesig' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt? Begründen Sie Ihre Antwort.

```
select /*+RULE*/ *
  from order_details
  where paketgroesse='riesig'
;
Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=HINT: RULE
1  0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS'
2  1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE)
```

RULE: Wenn Index da, dann nehme Index.

15. Erstellen Sie ein Histogramm der Tabelle order_details.

```
analyze table order_details
  compute statistics
  for columns quantity size 4
;
```

Histogramm wird nur mit 'size ...' erstellt.

16. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'riesig' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt? Welcher Optimierungsmodus wird eingesetzt? Begründen Sie Ihre Antwort.

```
select * from order_details
  where paketgroesse='riesig'
;
Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS'
2  1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE)
```

Index-Scan auf Grund des erstellten Histogramms.

17. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'mittel' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt? Welcher Optimierungsmodus wird eingesetzt? Begründen Sie Ihre Antwort.

```
select * from order_details
  where paketgroesse='mittel'
;
Ausführungsplan
----------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS'
2  1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE)
```

Index-Scan wurde verwendet. Full-Table-Scan könnte auch ausgewählt werden, wenn das Verhältnis Anzahl der Datensätze zur Anzahl unterschiedlicher Werte etwas anders wäre.

18. Erstellen Sie die Tabellen products und categories aus dem Skript.

```
start ~/1ZO-033/Northwind-Skript/products.txt
start ~/1ZO-033/Northwind-Skript/categories.txt
```

19. Erstellen Sie eine Abfrage, die Ihnen den Produktnamen und den Kategorienamen als Ergebnismenge zurückliefert. Um welchen Join handelte es sich hierbei und warum?

```
select p.productname,c.categoryname
  from products p, categories c
  where p.categoryid=c.categoryid
;
Ausführungsplan
------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   MERGE JOIN
2  1     SORT (JOIN)
3  2       TABLE ACCESS (FULL) OF 'CATEGORIES'
4  1     SORT (JOIN)
5  4       TABLE ACCESS (FULL) OF 'PRODUCTS'
```

20. Erstellen Sie einen Index auf der Spalte categoryid der Tabelle products. Führen Sie nun die Abfrage nochmals aus. Um welchen Join handelte es sich hierbei und warum?

```
create index id2 on products(categoryid);
select p.productname,c.categoryname
  from products p, categories c
  where p.categoryid=c.categoryid
;
Ausführungsplan
------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   TABLE ACCESS (BY INDEX ROWID) OF 'PRODUCTS'
2  1     NESTED LOOPS
3  2       TABLE ACCESS (FULL) OF 'CATEGORIES'
4  2       INDEX (RANGE SCAN) OF 'ID2' (NON-UNIQUE)
```

A.3.4 Stored Outlines

Im folgenden werden Sie Stored Outlines, also gespeicherte Ausführungspläne testen und einsetzen. Führen Sie hierfür folgende Schritte durch:

1. Schalten Sie Autotrace ein

```
set autotrace on explain;
```

2. Erstellen Sie die Tabelle order_details.

```
start ~/1ZO-033/Northwind-Skript/order_details.txt
```

3. Fügen Sie zur Tabelle order_details eine Spalte namens paketgroesse varchar(10) hinzu

```
alter table order_details add(paketgroesse varchar(10));
```

4. Geben Sie für 1/3 der Datensätze den Wert 'klein' in die Spalte Paketgroesse.

```
update order_details set paketgroesse=mod(rownum,3);
update order_details set paketgroesse='klein' where paketgroesse='1';
```

5. Geben Sie für 1/3 der Datensätze den Wert 'mittel' in die Spalte Paketgroesse.

```
update order_details set paketgroesse='mittel' where paketgroesse='2';
```

6. Geben Sie für 1/3 der Datensätze den Wert 'gross' in die Spalte Paketgroesse.

```
update order_details set paketgroesse='gross' where paketgroesse='0';
commit;
```

7. Erstellen Sie einen Index basierend auf der Spalte paketgroesse

```
create index id1 on order_details(paketgroesse);
```

8. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'klein' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt?

```
select * from order_details where paketgroesse='klein';

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS'
2  1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE)
```

9. Erstellen Sie eine Stored Outline für die eben erstellte Abfrage.

```
create or replace outline o1 for category cat1 on
  select * from order_details where paketgroesse='klein';
```

10. Aktivieren Sie diese Stored Outline Wie lautet die Anweisung hierfür?

```
alter session set use_stored_outlines=cat1;
```

11. Generieren Sie eine Statistik für die Tabelle order_details

```
analyze table order_details compute statistics;
```

12. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'klein' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt?

```
select * from order_details where paketgroesse='klein';

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE (Cost=9 Card=718 Bytes=14360)
1  0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS' (Cost=9 Card=718 Bytes=14360)
2  1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE) (Cost=2 Card=718)
```

13. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'mittel' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt?

```
select * from order_details where paketgroesse='mittel';

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE (Cost=3 Card=718 Bytes=14360)
1  0   TABLE ACCESS (FULL) OF 'ORDER_DETAILS' (Cost=3 Card=718 Bytes=14360)
```

14. Deaktivieren Sie diese Stored Outline.

```
alter session set use_stored_outlines=false;
Session wurde geändert.
```

15. Ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'klein' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt?

```
select * from order_details where paketgroesse='klein';

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE (Cost=3 Card=718 Bytes=14360)
1  0   TABLE ACCESS (FULL) OF 'ORDER_DETAILS' (Cost=3 Card=718 By tes=14360)
```

16. Aktivieren Sie diese Stored Outline erneut und ermitteln Sie die Datensätze aus order_details, deren Paketgröße 'klein' ist. Wird ein Index-Scan oder ein Full-Table-Scan durchgeführt?

```
alter session set use_stored_outlines=cat1;
select * from order_details where paketgroesse='klein';

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE (Cost=9 Card=718 Bytes=14360)
1  0   TABLE ACCESS (BY INDEX ROWID) OF 'ORDER_DETAILS' (Cost=9 Card=718 Bytes=14360)
2  1     INDEX (RANGE SCAN) OF 'ID1' (NON-UNIQUE) (Cost=2 Card=718)
```

A.3.5 Optimierungen

Spielen Sie die Datenbank aus einer Sicherung zurück.

Voraussetzung:

- Mehr als 1 MIO orders und mehr als 1 MIO products
- Tabelle suppliers (ganz normal ohne Verdoppelung)

Optimieren Sie folgende Anweisungen. Halten Sie Ihrer Vorgehensweise und die Ergebnisse einschließlich der Zeitangaben schriftlich fest.

1. Optimierung: Ermittlung der minimalen orderid

```
create index inx_oid
  on scott.orders(orderid) tablespace indx
;
analyze table scott.orders compute statistics
  for columns orderid size 16
;

select distinct min(orderid) from scott.orders;

MIN(ORDERID)
------------
       10248
Abgelaufen: 00:00:00.00

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (AGGREGATE)
2  1     INDEX (FULL SCAN (MIN/MAX)) OF 'INX_OID' (NON-UNIQUE)

-- Ausführungsplan speichern
create outline oid_min for category cat1 on
  select distinct min(orderid) from scott.orders
;
```

2. Optimierung: Wie viel Auslaufprodukte gibt es?

```
-- bitmap index erstellen
create bitmap index inx_disc
  on scott.products(discontinued) tablespace indx
;
-- analysieren
analyze table scott.products compute statistics
  for columns discontinued size 2
;
--  Fall 1
select count(*) from scott.products where discontinued=1;

  COUNT(*)
----------
    131072
```

```
Abgelaufen: 00:00:02.07

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (AGGREGATE)
2  1     TABLE ACCESS (FULL) OF 'PRODUCTS'

-- Fall 2
select /*+first_rows*/ count(*)
  from scott.products where discontinued = 1
;
  COUNT(*)
----------
    131072

Abgelaufen: 00:00:00.00

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=HINT: FIRST_ROWS (Cost=3 Card=1 Bytes=2)
1  0   SORT (AGGREGATE)
2  1     BITMAP CONVERSION (COUNT)
3  2       BITMAP INDEX (SINGLE VALUE) OF 'INX_DISC'
```

3. Optimierung: Welche Produkte fangen mit CH (Klein- oder Großschreibung egal) an?

```
create bitmap index id3 on products(productname);

analyze table products compute statistics for columns productname size 80;

select distinct productname from products
  where lower(productname) like 'ch%';

PRODUCTNAME
----------------------------------------
Chai
Chang
Chartreuse verte
Chef Anton's Cajun Seasoning
Chef Anton's Gumbo Mix
Chocolade

6 Zeilen ausgewählt.

Abgelaufen: 00:00:09.05

Ausführungsplan
----------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (UNIQUE)
2  1     TABLE ACCESS (FULL) OF 'PRODUCTS'

select /*+ index(id3) */distinct productname
  from products where lower(productname) like 'ch%'
;
```

```
PRODUCTNAME
----------------------------------------
Chai
Chang
Chartreuse verte
Chef Anton's Cajun Seasoning
Chef Anton's Gumbo Mix
Chocolade

6 Zeilen ausgewählt.

Abgelaufen: 00:00:00.01

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE (Cost=139 Card=77 Bytes=1309)
1  0   SORT (UNIQUE) (Cost=139 Card=77 Bytes=1309)
2  1     BITMAP INDEX (FAST FULL SCAN) OF 'ID3'
```

4. Optimierung: Wie teuer ist das Produkt namen Chai. Sie wissen nicht, ob dieses Produkt groß oder klein geschrieben ist. Optimieren Sie dieses Statement so weit es geht. Hier geht es um Geschwindigkeit um jeden Preis.

```
create index id34 on products(productname);
analyze index id34 validate structure;

select distinct productname,unitprice from products
  where lower(productname) = 'chai'
;

PRODUCTNAME             UNITPRICE
---------------------------------
Chai                           18

Abgelaufen: 00:00:03.09

Ausführungsplan
-------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (UNIQUE)
2  1     TABLE ACCESS (FULL) OF 'PRODUCTS'

select /*+ index(id34)*/ distinct productname,unitprice from products
where lower(productname) = 'chai';

PRODUCTNAME             UNITPRICE
---------------------------------
Chai                           18

Abgelaufen: 00:00:03.08

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE (Cost=1082 Card=8625 Bytes=301875)
1  0   SORT (UNIQUE) (Cost=1082 Card=8625 Bytes=301875)
2  1     TABLE ACCESS (FULL) OF 'PRODUCTS' (Cost=1017 Card=8625 Bytes=301875)
```

```
analyze index id34 delete statistics;
drop index id34;

create index fid34 on products(lower(productname));

select /*+ index(fid34)*/ distinct productname,unitprice from products
where lower(productname) = 'chai';

PRODUCTNAME             UNITPRICE
---------------------------------
Chai                           18

Abgelaufen: 00:00:02.02

Ausführungsplan
----------------------------------------------------------
0  SELECT STATEMENT Optimizer=CHOOSE (Cost=67 Card=8625 Bytes=301875)
1  0   SORT (UNIQUE) (Cost=67 Card=8625 Bytes=301875)
2  1     TABLE ACCESS (BY INDEX ROWID) OF 'PRODUCTS' (Cost=2 Card=8625 Bytes=301875)
3  2       INDEX (RANGE SCAN) OF 'FID34' (NON-UNIQUE) (Cost=1 Card=3450)

analyze index fid34 delete statistics;
drop index fid34;

create bitmap index bid34 on products(productname,unitprice);

select /*+ index(bid34)*/ distinct productname,unitprice
from products where lower(productname) = 'chai';

PRODUCTNAME             UNITPRICE
---------------------------------
Chai                           18

Abgelaufen: 00:00:00.01

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE (Cost=88 Card=8625 Bytes=301875)
1  0   SORT (UNIQUE) (Cost=88 Card=8625 Bytes=301875)
2  1     BITMAP INDEX (FAST FULL SCAN) OF 'BID34'

analyze index bid34 delete statistics;
drop index bid34;
```

5. Optimierung: Welche Produkte (productname und unitprice) werden von Lieferanten aus UK geliefert? Optimieren Sie hier um jeden Preis.

```
create index s_1 on suppliers(country, supplierid);
create bitmap index s_2 on products(productname, unitprice, supplierid);
select distinct productname, unitprice, country
  from products, suppliers
  where suppliers.supplierid(+)=products.supplierid and suppliers.country='UK'
;

PRODUCTNAME                              UNITPRICE COUNTRY
----------------------------------------------------------
Aniseed Syrup                                   10 UK
```

```
Chai                                          18 UK
Chang                                         19 UK
Scottish Longbreads                        12,5 UK
Sir Rodney's Marmalade                        81 UK
Sir Rodney's Scones                           10 UK
Teatime Chocolate Biscuits                   9,2 UK

7 Zeilen ausgewählt.

Abgelaufen: 00:00:01.07

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE (Cost=2393 Card=3376 Bytes=148544)
1  0 SORT (UNIQUE) (Cost=2393 Card=3376 Bytes=148544)
2  1   HASH JOIN (Cost=60 Card=142688 Bytes=6278272)
3  2     INDEX (RANGE SCAN) OF 'S_1' (NON-UNIQUE) (Cost=2 Card=2 Bytes=44)
4  2     BITMAP CONVERSION (TO ROWIDS)
5  4       BITMAP INDEX (FAST FULL SCAN) OF 'S_2'
```

6. Optimierung: Bei welchen Lieferanten ist der maximale Produktpreis größer als 200?

```
create bitmap index s_2 on products(supplierid, unitprice);
select supplierid
  from (select supplierid, max(unitprice) m1
  from products group by supplierid) where m1 > 200
;
SUPPLIERID
----------
        18

Abgelaufen: 00:00:02.08

Ausführungsplan
----------------------------------------------------------
0   SELECT STATEMENT Optimizer=CHOOSE (Cost=29 Card=2 Bytes=10)
1  0  FILTER
2  1    SORT (GROUP BY NOSORT) (Cost=29 Card=2 Bytes=10)
3  2      BITMAP CONVERSION (TO ROWIDS)
4  3        BITMAP INDEX (FULL SCAN) OF 'S_2'
```

7. Optimieren Sie folgende Abfrage: Wieviel Produkte haben an der zweiten Stelle ein h?

```
SQL> select count(distinct productname) from products where productname Like '_h%';

COUNT(DISTINCTPRODUCTNAME)
--------------------------
                         8
Abgelaufen: 00:00:00.02

Ausführungsplan
----------------------------------------------------------
0   SELECT STATEMENT Optimizer=CHOOSE (Cost=23 Card=1 Bytes=17)
1  0 SORT (GROUP BY)
2  1   BITMAP CONVERSION (TO ROWIDS)
```

```
3  2     BITMAP INDEX (FAST FULL SCAN) OF 'B2_IND'
```

A.3.6 Materialized Views

1. Geben Sie Scott die folgenden Berechtigungen:

```
grant CREATE SESSION to scott;
grant CREATE TABLE to scott;
grant CREATE MATERIALIZED VIEW to scott;
grant QUERY REWRITE to scott;
```

2. Melden Sie sich als scott mit dem Kennwort tiger an der Datenbank an.

```
connect scott/tiger@testdb29;
```

3. Erstellen Sie eine Tabelle namens bigtab, die auf den Spalten und Datensätzen der Tabelle all_objects basiert.

```
create table bigtab as select * from all_objects;
```

4. Verdoppeln Sie die Tabelle 5 mal. Wie viel Datensätze hat die Tabelle?

```
-- 5 mal
insert into bigtab select * from bigtab;
commit;

select count(*) from bigtab;

  COUNT(*)
----------
    751840
```

5. Analysieren Sie die Tabelle.

```
analyze table bigtab compute statistics;
```

6. Schreiben Sie eine Abfrage, die Ihnen die Anzahl der Datensätze je owner anzeigt. Messen Sie die Zeit für die Ausführung dieser Abfrage und halten Sie Ihr Ergebnis schriftlich fest.

```
OWNER        COUNT(*)
--------------------
CTXSYS           6264
ELAN             1272
HR                816
MDSYS            5640
ODM              9768
ODM_MTR           288

set timing on;
```

7. Führen Sie folgende Befehle aus.

```
alter session set query_rewrite_enabled=true;
alter session set query_rewrite_integrity=enforced;
```

8. Erstellen Sie folgende Materialized View:

```
create materialized view mv_bigtab
   build immediate
   refresh on commit
   enable query rewrite
 as
   select owner, count(*) from bigtab group by owner
;
Materialized View wurde erstellt.

Abgelaufen: 00:00:12.08
```

9. Führen Sie folgende Abfrage aus. Messen Sie bitte die Zeit:

```
select owner,count(*) from bigtab group by owner;

Abgelaufen: 00:00:00.00
```

10. Was sind Ihre Schlussfolgerungen?

Es wurde die Materialized View verwendet. Dies ist eine sehr optimale Methode.

11. Führen Sie folgende Anweisung aus:

```
alter session set query_rewrite_enabled=false;
```

12. Was macht die Anweisung?

Die Materialized View kann so nicht mehr verwendet werden.

13. Führen Sie folgende Abfrage aus. Messen Sie bitte die Zeit:

```
select owner, count(*) from bigtab group by owner;

Abgelaufen: 00:00:07.05
```

14. Was sind Ihre Schlussfolgerungen?

Die Materialized View wurde nicht mehr verwendet.

15. Führen Sie folgende Anweisung aus:

```
alter session set query_rewrite_enabled = true;
```

16. Was macht die Anweisung?

Ermöglicht die Verwendung der Materialized View.

17. Fügen Sie einen Datensatz in die Tabelle bigtab hinzu.

```
insert into bigtab
         (OWNER, OBJECT_NAME, OBJECT_ID, CREATED, LAST_DDL_TIME)
  values ('Hans','bla',1,sysdate,sysdate)
;
```

18. Schließen Sie das Einfügen mit commit auch ab.

```
commit;
```

19. Führen Sie folgende Abfrage aus.

```
select owner, count(*) from bigtab group by owner;

Abgelaufen: 00:00:00.00
```

20. Was sind Ihre Schlussfolgerungen? Ist der neue Datensatz mit enthalten und wurde der materialisierte View benutzt?

Mit commit wird die Materialized View aktualisiert.

21. Testen Sie selbständig, ob das auch für das Löschen gilt.

```
delete bigtab where OWNER='Hans';
commit;

select owner, count(*) from bigtab group by owner;

OWNER                          COUNT(*)
------------------------------ ----------
CTXSYS                               2048
MDSYS                                5376
...
Abgelaufen: 00:00:00.00
```

Auch beim Löschen wird die Materialized View aktualisiert.

22. Schreiben Sie die Abfrage so um, dass nur die Datensätze angezeigt werden, bei denen zum Owner mehr als 1000 Datensätze gehören. Führen Sie die Abfrage aus und beobachten Sie, ob der materialisierte View benutzt wird.

```
start /oracle/ora92/rdbms/admin/utlxplan.sql
set autotrace on explain;

select owner, count(*) from bigtab group by owner having count(*) > 1000;

OWNER                          COUNT(*)
------------------------------ ----------
CTXSYS                               2048
MDSYS                                5376
ODM                                  7904
OLAPSYS                              4256
ORDSYS                              28896
PUBLIC                             369280
SYS                                322528
WKSYS                                3776
WMSYS                                1824
```

```
XDB                                 5152

10 Zeilen ausgewählt.

Abgelaufen: 00:00:00.00

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE (Cost=2 Card=4 Bytes=120)
1  0   TABLE ACCESS (FULL) OF 'MV_BIGTAB' (Cost=2 Card=4 Bytes=120)
```

Es wird die Materialized View verwendet, obwohl das Matching nicht exakt ist. Bei HAVING bringen Materialized Views Vorteile.

23. Löschen Sie die Materialized View wieder.

```
drop materialized view name;
```

Zusatzaufgaben

1. Erstellen Sie eine Tabelle namens lehrer und tragen Sie dort 5 Lehrer ein. Bestätigen Sie das mit commit.

```
create table lehrer
  (
    Name varchar2(29)
  )
;
insert into lehrer values ('Meier');
insert into lehrer values ('Müller');
insert into lehrer values ('Schmidt');
insert into lehrer values ('Schulz');
insert into lehrer values ('Meyer');
commit;
```

2. Erstellen Sie einen materialized view namens mv1, der die Anzahl der Datensätze der Tabelle lehrer ausgibt.

```
create materialized view mv1
   build immediate
   refresh on commit
   enable query rewrite
 as
   select count(*) from lehrer
;
```

Hinweis (ohne count):

```
SQL> create materialized view mv1
  2     build immediate
  3     refresh on commit
  4     enable query rewrite
  5   as
  6     select * from lehrer
  7  ;
   select * from lehrer
                 *
FEHLER in Zeile 6:
ORA-12014: Tabelle 'LEHRER' enthält kein Primärschlüssel-Constraint
```

3. Schreiben Sie eine Abfrage basierend auf der Tabelle lehrer, welche die Anzahl der Datensätze ermittelt und beobachten Sie, ob die Materialized View benutzt wird.

```
select count(*) from lehrer;

  COUNT(*)
----------
         5

Abgelaufen: 00:00:00.00

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (AGGREGATE)
2  1     TABLE ACCESS (FULL) OF 'LEHRER'
```

Die Materialized Views wurde nicht verwendet, da der CBO nicht aktiv ist (kein ANALYZE).

4. Fügen Sie einen Datensatz zur Tabelle lehrer hinzu.

```
insert into lehrer values ('Kohl');
commit;
```

5. Schreiben Sie eine Abfrage basierend auf der Tabelle lehrer, welche die Anzahl der Datensätze ermittelt und beobachten Sie, ob der Materialized View benutzt wird.

```
select count(*) from lehrer;
  COUNT(*)
----------
         6

Abgelaufen: 00:00:00.00

Ausführungsplan
----------------------------------------------------------
0    SELECT STATEMENT Optimizer=CHOOSE
1  0   SORT (AGGREGATE)
2  1     TABLE ACCESS (FULL) OF 'LEHRER'
```

A.3.7 Ressourcen-Manager

Teil 1

Erstellen Sie folgenden Ressource-Manager-Plan per Anweisung. Aktivieren Sie ihn anschließend. Erstellen Sie auch folgende Benutzer und ordnen Sie diese den entsprechenden Gruppen zu.
OLTP_Group: OLTP_1
BATCH_Group: Batch_1
ADHOC_Group: Adhoc_1

Ressourcen Gruppen	Level 1	Level 2	Level 3	Parallel Degree Limit
OLTP_GROUP	20%	0%	0%	0
BATCH_GROUP	0%	80%	0%	10
ADHOC_GROUP	0%	20%	0%	50
OTHER_GROUP	0%	0%	100%	0

User erstellen

```
create user OLTP_1 identified by OLTP_1;
create user Batch_1 identified by Batch_1;
create user Adhoc_1 identified by Adhoc_1;
```

Pending-Area neu erstellen.

```
exec dbms_resource_manager.clear_pending_area();
exec dbms_resource_manager.create_pending_area();
```

NIGHT_PLAN erstellen

```
exec dbms_resource_manager.create_plan(plan => 'NIGHT_PLAN',comment=>'Mein erster Plan');
```

Gruppen erstellen

```
exec dbms_resource_manager.create_consumer_group
  (consumer_group => 'OLTP_Group',comment=>'Meine erste Gruppe')
;
exec dbms_resource_manager.create_consumer_group
  (consumer_group => 'BATCH_Group',comment=>'Meine zweite Gruppe')
;
exec dbms_resource_manager.create_consumer_group
  (consumer_group => 'ADHOC_Group',comment=>'Meine dritte Gruppe')
;
```

Resources

```
exec dbms_resource_manager.create_plan_directive
  (plan => 'NIGHT_PLAN', group_or_subplan => 'OLTP_Group',
   cpu_p1 => 20, parallel_degree_limit_p1 =>  0, comment=>'')
;
exec dbms_resource_manager.create_plan_directive
  (plan => 'NIGHT_PLAN', group_or_subplan => 'BATCH_Group',
   cpu_p2 => 80, parallel_degree_limit_p1 => 10, comment=>'')
;
exec dbms_resource_manager.create_plan_directive
  (plan => 'NIGHT_PLAN', group_or_subplan => 'ADHOC_Group',
   cpu_p2 => 20, parallel_degree_limit_p1 => 5, comment=>'')
;
exec dbms_resource_manager.create_plan_directive
  (plan => 'NIGHT_PLAN', group_or_subplan => 'OTHER_GROUPS',
   cpu_p3 => 100, parallel_degree_limit_p1 =>  0, comment=>'')
;
```

Submit

```
exec dbms_resource_manager.submit_pending_area();
exec dbms_resource_manager.create_pending_area();
```

Berechtigung den Users zuweisen

```
exec dbms_resource_manager_privs.grant_switch_consumer_group
  (grantee_name => 'OLTP_1',consumer_group => 'OLTP_Group', grant_option => FALSE)
;
exec dbms_resource_manager_privs.grant_switch_consumer_group
  (grantee_name => 'Batch_1',consumer_group => 'BATCH_Group', grant_option => FALSE)
;
exec dbms_resource_manager_privs.grant_switch_consumer_group
  (grantee_name => 'Adhoc_1',consumer_group => 'ADHOC_Group', grant_option => FALSE)
;
```

User zuweisen

```
exec dbms_resource_manager.set_initial_consumer_group
  (user => 'OLTP_1',consumer_group => 'OLTP_Group')
;
exec dbms_resource_manager.set_initial_consumer_group
  (user => 'Batch_1',consumer_group => 'BATCH_Group')
;
exec dbms_resource_manager.set_initial_consumer_group
  (user => 'Adhoc_1',consumer_group => 'ADHOC_Group')
;
```

Plan aktivieren

```
alter system set resource_manager_plan = 'NIGHT_PLAN';
```

Teil 2

Verändern Sie den Plan so, dass für alle anderen (also other_groups) alle Abfragen, die länger als 5 Sekunden dauern, geblockt werden. Überprüfen Sie Ihr Ergebnis anhand eines geeigneten Szenarios

```
exec dbms_resource_manager.clear_pending_area();
exec dbms_resource_manager.create_pending_area();
```

Update

```
exec dbms_resource_manager.update_plan_directive
  (plan => 'NIGHT_PLAN', group_or_subplan => 'OTHER_GROUPS', new_max_est_exec_time => 5)
;
```

Submit

```
exec dbms_resource_manager.SUBMIT_pending_area();
```

Teil 3

Verändern Sie den Plan so, dass für adhoc_user alle Abfragen, die länger als 20 Sekunden dauern, geblockt werden. Außerdem sollten diese User nicht mehr als 6 paralelle Sessions geöffnet haben können. Überprüfen Sie Ihr Ergebnis anhand eines geeigneten Szenarios.

```
exec dbms_resource_manager.clear_pending_area();
exec dbms_resource_manager.create_pending_area();
```

Update

```
exec dbms_resource_manager.update_plan_directive
  (plan => 'NIGHT_PLAN', group_or_subplan => 'ADHOC_Group',
   new_max_est_exec_time => 20, new_parallel_degree_limit_p1 => 6)
;
```

Submit

```
exec dbms_resource_manager.SUBMIT_pending_area();
```

A.3.8 Buffer-Cache

In der folgenden Übung werden Sie den Buffer-Cache einrichten und tunen. Voraussetzung hierfür ist Ihre Datenbank und ein funktionierendes SPFILE. Sollte dies noch nicht erfüllt sein, so unternehmen Sie alle notwendigen Schritte hierfür. Führen Sie, um Ihre Datenbank wieder im Fehlerfall wieder herstellen zu können, eine Offline-Sicherung Ihrer Datenbank durch. Die Ergebnisse können durchaus variieren.

1. Erstellen Sie eine Tabelle abteilung mit einer Spalte S1 char(1000) und zehn Datensätzen (von 'AAA' bis 'JJJ')

```
create table abteilung (s1 char(1000));
insert into abteilung values('AAA');
insert into abteilung values('BBB');
insert into abteilung values('CCC');
insert into abteilung values('DDD');
insert into abteilung values('EEE');
insert into abteilung values('FFF');
insert into abteilung values('GGG');
insert into abteilung values('HHH');
insert into abteilung values('III');
insert into abteilung values('JJJ');
```

2. Verdoppeln Sie die Tabelle 10 mal.

```
-- 10 x
insert into abteilung select * from abteilung;
commit;
```

3. Wie viel Datensätze haben Sie nun?

```
select count(*) from abteilung;
  COUNT(*)
----------
     10240
```

4. Stellen Sie sicher, dass die gecachten Blöcke niemals ausgelagert werden. Ermitteln Sie hierfür die Größe der Tabelle abteilung. Erstellen Sie gemäß dieser Größe den entsprechenden Buffer-Pool. Stellen Sie sicher, dass die Tabelle in den Keep-Pool geschrieben wird.

```
select sum(bytes)
  from dba_segments
  where owner = 'SYS'
    and segment_name = 'ABTEILUNG'
;
SUM(BYTES)
----------
  12582912
```

= 12 MByte

```
alter system set DB_KEEP_CACHE_SIZE=16M;
-- Anzeige:
select 1, name, to_number(value/1024/1024) value
  from v$parameter
  where upper(name) like 'DB%CACHE_SIZE'
    or
       upper(name)
       in ('SHARED_POOL_SIZE','LARGE_POOL_SIZE','JAVA_POOL_SIZE','LOG_BUFFER')
     union
       select 1, '+ 1MB', 1 from dual order by 2
;
-- Tabelle in den Keep-Pool
ALTER TABLE ABTEILUNG STORAGE (buffer_pool KEEP);
```

5. Führen Sie mehrfach eine Abfrage der Tabelle Abteilung durch und beobachten Sie jeweils die Hit Ratio für den Keep-Pool.

```
-- Abfrage (mehrmals)
select count(*) from abteilung where s1='AAA';

-- Abfrage Hit Ratio:
select 1-value/
      (
        select sum(value)
          from v$sysstat
          where lower(name) in ('consistent gets','db block gets')
      )
  from v$sysstat
  where lower(name) in ('physical reads')
;
,959986246
,960109863
,960304683
```

Hit Ratio steigt.

6. Beobachten Sie ebenfalls die Werte in der Tabelle v$buffer_pool_statistics (PHYSICAL_READS, DB_BLOCK_GETS, CONSISTENT_GETS).

```
select name,
       100-round((physical_reads/(db_block_gets+consistent_gets))*100,2)
        as Hit_Ratio
  from v$buffer_pool_statistics
;
FEHLER in Zeile 1:
ORA-01476: Divisor ist Null
```

Da KEEP gleich 0 ist. Ggf. ANALYZE ??

```
select name, PHYSICAL_READS, DB_BLOCK_GETS, CONSISTENT_GETS
  from v$buffer_pool_statistics
;
NAME           PHYSICAL_READS DB_BLOCK_GETS CONSISTENT_GETS
-------------- -------------- ------------- ---------------
KEEP                        0             0               0
DEFAULT                 17251        357396          202093
```

7. Löschen Sie die Tabelle Abteilung.

```
drop table abteilung;
```

8. Führen Sie die gleichen Tests für den Recycle-Pool durch. Halten Sie auch hier Ihre Ergebnisse schriftlich fest:

Bis 4. sind die Schritte identisch.

```
alter system set DB_KEEP_CACHE_SIZE=0M;
alter system set db_recycle_cache_size=16M;
ALTER TABLE ABTEILUNG STORAGE (buffer_pool recycle);


-- Abfrage (mehrmals)
select count(*) from abteilung where s1='AAA';

-- Abfrage Hit Ratio:
select 1-value/
       (
        select sum(value)
          from v$sysstat
          where lower(name) in ('consistent gets','db block gets')
       )
  from v$sysstat
  where lower(name) in ('physical reads')
;
,958975396
,959156127
,959268817
,959413019


select name,
       100-round((physical_reads/(db_block_gets+consistent_gets))*100,2)
         as Hit_Ratio
  from v$buffer_pool_statistics
;
FEHLER in Zeile 1:
ORA-01476: Divisor ist Null
```

```
select name, PHYSICAL_READS, DB_BLOCK_GETS, CONSISTENT_GETS
  from v$buffer_pool_statistics
;
NAME                 PHYSICAL_READS DB_BLOCK_GETS CONSISTENT_GETS
-------------------- -------------- ------------- ---------------
RECYCLE                           0             0               0
DEFAULT                       19180        394619          223113
```

9. Setzen Sie den buffer_cache auf 16 MB.

```
alter system set db_cache_size=16M;
```

10. Erzeugen Sie eine Tabelle angestellte auf Basis der Tabelle scott.emp

```
create table angestellte as select * from scott.emp;
```

11. Vergrößern Sie die Tabelle so lange, bis diese 32MB groß ist.

```
insert into ANGESTELLTE select * from ANGESTELLTE;
commit;
insert into ANGESTELLTE select * from ANGESTELLTE;
commit;
-- ...
select sum(bytes)
  from dba_segments
  where owner = 'SYS'
    and segment_name = 'ANGESTELLTE'
;
SUM(BYTES)
----------
  45088768
```

12. Fragen Sie die Tabelle ab.

```
select count(*) from ANGESTELLTE;
  COUNT(*)
----------
    917504
```

13. Schauen Sie sich das Hit Ratio an. Was passiert und warum?

```
select 1-value/
       (
         select sum(value)
           from v$sysstat
           where lower(name) in ('consistent gets','db block gets')
        )
  from v$sysstat
  where lower(name) in ('physical reads')
;
,938784339
,93226288
,92621924
,919666046
```

Hit Ratio nimmt ab.

14. Setzen Sie den Buffer_Cache auf 38 MB.

```
alter system set db_cache_size=38M;
```

15. Fragen Sie die Tabelle Angestellte ab.

```
select count(*) from ANGESTELLTE;
```

16. Schauen Sie sich das Hit Ratio an. Was passiert und warum?

```
select 1-value/
       (
         select sum(value)
           from v$sysstat
           where lower(name) in ('consistent gets','db block gets')
        )
  from v$sysstat
  where lower(name) in ('physical reads')
;
,697806485
```

17. Fragen Sie die Tabelle Angestellte ab. Da Sie die Abfrage gleich wiederholen möchten, sollten die Blöcke nicht gleich wieder aus dem cache rausfliegen. Was müssen Sie tun?

```
,691264487
,682978723
```

18. Stellen Sie sicher, dass Full-Table-Scans der Tabelle Angestellte immer am MRU-Ende angefügt werden. Was müssen Sie hierfür tun?

```
select /*+ CACHE(ANGESTELLTE) */ count(*)
  from ANGESTELLTE
;
,739961835
```

19. Probieren Sie den Cache-Advise aus.

```
,727582741
```

A.3.9 Kleine Optimierungsübung

Welche Produkte (productname) je Lieferant sind die preiswertesten?

```
grant CREATE SESSION to scott;
grant CREATE TABLE to scott;
grant CREATE MATERIALIZED VIEW to scott;
grant QUERY REWRITE to scott;
```

Da Materialized View unter sys nicht möglich sind, loggen wir uns als scott ein.

```
connect scott/tiger@testdb29;


start ~/1ZO-033/Northwind-Skript/suppliers.txt
start ~/1ZO-033/Northwind-Skript/products.txt
set timing on;


alter session set query_rewrite_enabled=true;
alter session set query_rewrite_integrity=enforced;
analyze table products compute statistics;
analyze table suppliers compute statistics;


select q.companyname, p.productname, q.Preis
  from products p,
   (
    select s.companyname as companyname,s.supplierid, min(p.unitprice) as Preis
      from products p, suppliers s
      where p.supplierid = s.supplierid
      group by s.companyname, s.supplierid
   ) q
   where p.supplierid = q.supplierid
    and q.Preis = p.unitprice
;
30 Zeilen ausgewählt.

Abgelaufen: 00:00:00.03


create materialized view mv_min_preise
   build immediate
   enable query rewrite
 as
   select q.companyname, p.productname, q.Preis
     from products p,
      (
       select s.companyname as companyname,s.supplierid, min(p.unitprice) as Preis
         from products p, suppliers s
         where p.supplierid = s.supplierid
         group by s.companyname, s.supplierid
      ) q
     where p.supplierid = q.supplierid
       and q.Preis = p.unitprice
;
-- refresh on commit nicht möglich ??


select * from mv_min_preise;
30 Zeilen ausgewählt.

Abgelaufen: 00:00:00.01


select q.companyname, p.productname, q.Preis
  from products p,
   (
    select s.companyname as companyname,s.supplierid, min(p.unitprice) as Preis
      from products p, suppliers s
      where p.supplierid = s.supplierid
      group by s.companyname, s.supplierid
```

```
   ) q
   where p.supplierid = q.supplierid
    and q.Preis = p.unitprice
;

Ausführungsplan
----------------------------------------------------------
0   SELECT STATEMENT Optimizer=CHOOSE (Cost=2 Card=82 Bytes=4674)
1 0   TABLE ACCESS (FULL) OF 'MV_MIN_PREISE' (Cost=2 Card=82 Bytes=4674)
```

In der Unix-Shell:

```
cd /oracle/admin/testdb29/udump/
tkprof testdb29_ora_1792.trc pinguin.trc
```

A.3.10 Undo / Rollback Segments

Im folgenden werden Sie das Undo-Management von Oracle etwas genauer kennenlernen.

1. Stellen Sie das Undo-Management auf Manuell um.

```
select * from v$tablespace;
...
1 UNDOTBS1  YES
...
```

Undo-Tablespace ist vorhanden.

```
alter system set undo_management=manual scope=spfile;
shutdown immediate;
startup;
```

2. Erstellen Sie ein neues Undo-Tablespace namens undots02. Alle weiteren Parameter können Sie frei wählen. Schalten Sie diesen Online.

```
CREATE UNDO TABLESPACE undots02
  DATAFILE '/oracle/oradata/testdb29/undots02.dbf'
  SIZE 100M AUTOEXTEND ON
;
Tablespace wurde angelegt.

alter tablespace undots02 online;
```

3. Erstellen Sie in diesem Tablespace 5 Rollbacksegmente (rbs01 bis rbs05) und schalten Sie diese Online.

```
CREATE ROLLBACK SEGMENT rbs01 TABLESPACE undots02;
CREATE ROLLBACK SEGMENT rbs02 TABLESPACE undots02;
CREATE ROLLBACK SEGMENT rbs03 TABLESPACE undots02;
CREATE ROLLBACK SEGMENT rbs04 TABLESPACE undots02;
CREATE ROLLBACK SEGMENT rbs05 TABLESPACE undots02;
alter rollback segment rbs01 online;
alter rollback segment rbs02 online;
alter rollback segment rbs03 online;
alter rollback segment rbs04 online;
alter rollback segment rbs05 online;
```

4. Ändern Sie den Preis aller Produkte der Tabelle products um 10% nach oben.
Die Undo-Informationen sollten im Rollback-Segment rbs01 erstellt werden.

```
set transaction use rollback segment rbs01;
start ~/1ZO-033/Northwind-Skript/products.txt
```

5. Versuchen Sie, die Szenarien 1 bis 5 nachzuspielen.

```
GRANT select ON products TO scott;
```

Szenario 1

```
-- Session 1:
connect "sys/sys@testdb29 as sysdba";
-- Session 2:
connect scott/tiger@testdb29;
-- Session 1:
update products set unitprice = unitprice*1.1;
77 Zeilen wurden aktualisiert.
-- Session 2:
select productname, unitprice
  from sys.products
  where productid = '75'
;
PRODUCTNAME                              UNITPRICE
---------------------------------------- ----------
Rhonbräu Klosterbier                           9,38

-- Session 1:
commit;
-- Session 2:
select productname, unitprice
  from sys.products
  where productid = '75'
;
PRODUCTNAME                              UNITPRICE
---------------------------------------- ----------
Rhonbräu Klosterbier                          10,32
```

Szenario 2 und 3

```
-- Session 1:
delete from products where productid=77;
select productname, unitprice from sys.products where productid = '77';
Es wurden keine Zeilen ausgewählt.
-- Session 2:
select productname, unitprice from sys.products where productid = '77';
PRODUCTNAME                              UNITPRICE
---------------------------------------- ----------
Original Frankfurter grüne So¯e                17,3
-- Session 1:
commit;
-- Session 2:
select productname, unitprice from sys.products where productid = '77';
Es wurden keine Zeilen ausgewählt.
```

Szenario 4

```
-- Session 2:
set transaction read only;
select productname, unitprice from sys.products where productid = '75';
PRODUCTNAME                              UNITPRICE
---------------------------------------- ----------
Rhönbräu Klosterbier                          10,32
-- Session 1:
update products set unitprice = unitprice*1.1;
commit;
-- Session 2:
set transaction read only;
select productname, unitprice from sys.products where productid = '75';
PRODUCTNAME                              UNITPRICE
---------------------------------------- ----------
Rhönbräu Klosterbier                          10,32
-- Session 1:
update products set unitprice = unitprice*1.1;
commit;
select productname, unitprice from sys.products where productid = '75';
PRODUCTNAME                              UNITPRICE
---------------------------------------- ----------
Rhönbräu Klosterbier                          12,49
-- Session 2:
select productname, unitprice from sys.products where productid = '75';
PRODUCTNAME                              UNITPRICE
---------------------------------------- ----------
Rhönbräu Klosterbier                          10,32
--
-- Ein commit beendet set transaction read only:
commit;
-- Session 1:
update products set unitprice = unitprice*1.1;
commit;
select productname, unitprice from sys.products where productid = '75';
PRODUCTNAME                              UNITPRICE
---------------------------------------- ----------
Rhönbräu Klosterbier                          13,74
-- Session 2:
select productname, unitprice from sys.products where productid = '75';
PRODUCTNAME                              UNITPRICE
---------------------------------------- ----------
Rhönbräu Klosterbier                          13,74
```

Szenario 5:

```
-- Session 2:
set transaction read only;
-- Session 1:
-- 12 mal
insert into sys.products select * from sys.products;
commit;
delete from sys.products where productid=75;
4096 Zeilen wurden gelöscht.
insert into sys.products select * from sys.products;
...
```

```
-- Session 2:
select productname, unitprice from sys.products where productid = '75';
select productname, unitprice from sys.products where productid = '75'
*
FEHLER in Zeile 1:
ORA-01555: Snapshot zu alt: Rollback-Segmentnummer 24 namens "RBS04" ist zu klein.
```

A.3.11 StatsPack

1. Löschen Sie alle großen Tabellen.

```
drop table ...;
```

2. Erstellen Sie die Datenbank northwind vom Skript.

```
start ~/1ZO-033/Northwind-Skript/alles.txt
```

3. Führen Sie alle notwendigen Schritte aus, um StatsPack zu installieren.

```
start /oracle/ora92/rdbms/admin/spcreate.sql
...
Geben Sie einen Wert für perfstat_password ein: pinguin
...
Geben Sie einen Wert für default_tablespace ein: USERS
...
Geben Sie einen Wert für temporary_tablespace ein: TEMP
...
NOTE:
SPCPKG complete. Please check spcpkg.lis for any errors.
```

4. Erstellen Sie folgende Abfragen, speichern Sie diese, um diese Abfragen später ausführen zu können.

Wie viele Kunden haben Getränke bestellt?

```
select distinct companyname
  from
        customers cus,
        orders ord,
        order_details ode,
        products pro,
        categories cat
  where
        cus.customerid   = ord.customerid
    and ode.orderid      = ord.orderid
    and pro.productid    = ode.productid
    and cat.categoryid   = pro.categoryid
    and cat.Categoryname = 'Beverages'
;
```

Wie heißt der älteste Mitarbeiter?

```
select lastname,firstname,birthdate
  from employees
  where birthdate in
  (select min(birthdate) from employees)
;
```

Welche Produkte liefern deutsche Lieferanten?

```
select distinct pro.productname
  from
    suppliers sup, products pro
  where
    sup.supplierid = pro.supplierid
    and country = 'Germany'
;
```

5. Erstellen Sie den ersten Snapshot.

```
sqlplus perfstat/pinguin@testdb29
execute statspack.snap
```

6. Führen Sie die Abfragen aus.

s.o.

7. Erstellen Sie den zweiten Snapshot

```
sqlplus perfstat/pinguin@testdb29
execute statspack.snap
```

8. Erstellen Sie den Bericht

```
select snap_id, snap_time from stats$snapshot;

   SNAP_ID SNAP_TIM
---------- --------
         1 05.03.04
         2 05.03.04

start /oracle/ora92/rdbms/admin/spreport.sql
...
Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Geben Sie einen Wert für begin_snap ein: 1
Begin Snapshot Id specified: 1
Geben Sie einen Wert für end_snap ein: 2
...
Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is sp_1_2.  To use this name,
press <return> to continue, otherwise enter an alternative.
Geben Sie einen Wert für report_name ein: /tmp/sp_report_1_2
```

9. Schauen Sie sich den Bericht im Betriebssystem an.

```
less /tmp/sp_report_1_2
```

A.3.12 Index-Cluster

Im folgenden werden Sie einen Index-Cluster kennen lernen und praktisch erstellen.

1. Führen Sie folgende Anweisung aus, um den Cluster zu erstellen.

```
CREATE CLUSTER personnel (department_number NUMBER(2)) SIZE 512;
```

2. Was bedeutet Size 512?

SIZE sollte in der Größe eines "Datensatz"definiert werden.

3. Erstellen Sie eine Tabelle emp und ordnen diese dem Cluster zu.

```
CREATE TABLE emp
  (
     empno     NUMBER        PRIMARY KEY,
     ename     VARCHAR2(10)  NOT NULL CHECK (ename = UPPER(ename)),
     job       VARCHAR2(9),
     mgr       NUMBER        ,
     hiredate  DATE          ,
     sal       NUMBER(10,2)  CHECK (sal > 500),
     comm      NUMBER(9,0)   DEFAULT NULL,
     deptno    NUMBER(2)      NOT NULL
   )
   CLUSTER personnel (deptno)
;
```

4. Erstellen Sie eine Tabelle dept und ordnen diese dem Cluster zu.

```
CREATE TABLE dept
  (
    deptno  NUMBER(2),
    dname   VARCHAR2(9),
    loc     VARCHAR2(9)
  )
  CLUSTER personnel (deptno)
;
```

5. Versuchen Sie einen Datensatz in eine der beiden Tabellen hinzuzufügen.

```
insert into dept values (1,'Head','Berlin');
            *
FEHLER in Zeile 1:
ORA-02032: Cluster-Tabellen sind erst nach Aufbau des Cluster-Indexes zu
verwenden
```

6. Erstellen Sie nun den Index-Cluster.

```
CREATE INDEX idx_personnel ON CLUSTER personnel;
```

7. Versuchen Sie einen Datensatz in eine der beiden Tabellen hinzuzufügen.

```
insert into dept values (1,'Head','Berlin');
1 Zeile wurde erstellt.
commit;
```

8. Veranlassen Sie, dass die Tabellen products_cl und suppliers_cl in einem Index-Cluster zusammengefasst werden. Begründen Sie Ihre Wahl für die Size.

Berechnung der Größe von SIZE:

```
describe products;
 Name                                      Null?    Typ
 ----------------------------------------- -------- -------------

 PRODUCTID                                 NOT NULL NUMBER(38)
 PRODUCTNAME                               NOT NULL VARCHAR2(40)
 SUPPLIERID                                         NUMBER(38)
 CATEGORYID                                         NUMBER(38)
 QUANTITYPERUNIT                                    VARCHAR2(20)
 UNITPRICE                                          NUMBER(20,2)
 UNITSINSTOCK                                       NUMBER(38)
 UNITSONORDER                                       NUMBER(38)
 REORDERLEVEL                                       NUMBER(38)
 DISCONTINUED                              NOT NULL NUMBER(38)

Spaltenbreite gesamt: 346

describe suppliers;
 Name                                      Null?    Typ
 ----------------------------------------- -------- --------------

 SUPPLIERID                                NOT NULL NUMBER(38)
 COMPANYNAME                               NOT NULL VARCHAR2(40)
 CONTACTNAME                                        VARCHAR2(30)
 CONTACTTITLE                                       VARCHAR2(30)
 ADDRESS                                            VARCHAR2(60)
 CITY                                               VARCHAR2(15)
 REGION                                             VARCHAR2(15)
 POSTALCODE                                         VARCHAR2(10)
 COUNTRY                                            VARCHAR2(15)
 PHONE                                              VARCHAR2(24)
 FAX                                                VARCHAR2(24)
 HOMEPAGE                                           VARCHAR2(255)
```

Spaltenbreite gesamt: 556

Ermitteln des Durchschnittswertes der 1-n-Beziehung:

```
select avg(count(*)) from products group by supplierid order by 1;

AVG(COUNT(*))
-------------
   2,65517241
```

Spaltenbreite gesamt der ersten Tabelle minus Spaltenbreite der gemeinsamen Spalte (SUPPLIERID) multipliziert mit dem ermittelten Durchschnitt. Die Spaltenbreite der Tabelle zwei wird dazu addiert und zur Sicherheit wird alles mit 20% multipliziert.

((346 - 38) * 2,65517241 + 556) * 1,2 = 1648,551722736

Das Ergebnis wird aufgerundet.

Cluster erstellen:

```
CREATE CLUSTER prod_supp_cl (SUPPLIERID NUMBER(38)) SIZE 1700;
```

Cluster Index erstellen:

```
CREATE INDEX idx_prod_supp_cl ON CLUSTER prod_supp_cl;
```

Erste Tabellen erstellen
Beachte: CategoryID ist als int zu definieren und nicht als number.

```
CREATE TABLE products_cl
  (
    ProductID      int   NOT NULL ,
    ProductName    varchar2  (40) NOT NULL ,
    SupplierID     number(38)  NULL ,
    CategoryID     int  NULL ,
    QuantityPerUnit   varchar2  (20) NULL ,
    UnitPrice      number(20,2)  NULL ,
    UnitsInStock   int  NULL ,
    UnitsOnOrder   int  NULL ,
    ReorderLevel   int  NULL ,
    Discontinued   int  NOT NULL
  )
  CLUSTER prod_supp_cl(SUPPLIERID)
;
insert into products_cl select * from products;
```

Zweite Tabelle
Beachte: CategoryID ist als int zu definieren und nicht als number. Die Quelltabelle muss daher angepasst werden:

```
alter table suppliers modify (SUPPLIERID number(38));

CREATE TABLE suppliers_cl
  CLUSTER prod_supp_cl(SUPPLIERID)
  as select * from suppliers
;
```

Im folgenden werden Sie eine IOT erstellen. Führen Sie hierfür folgende Schritte durch: Führen Sie folgende Anweisung aus:

```
CREATE TABLE my_iot
  (partno number,name varchar2(20),CONSTRAINT pk_my_iot PRIMARY KEY (partno))
  ORGANIZATION INDEX TABLESPACE users including name
  OVERFLOW TABLESPACE users
;
```

Literaturverzeichnis

[1] Stefan Hietel. *dama.go GmbH.*
http://www.damago.de.

[2] Robert Warnke. *My Website.*
http://rowa.giso.de.

[3] Robert Warnke. *My Website, Oracle.*
http://rowa.giso.de/oracle.

[4] Website. *mySQL-ODBC-Treiber.*
http://www.mysql.com/get/Downloads/MyODBC3/MyODBC-3.51.06.exe/from/pick.

[5] Website. *Oracle.*
http://www.oracle.com.

[6] Website. *Oracle Certification.*
http://www.oracle.com/education/certification.

[7] Website. *Oracle Download.*
http://otn.oracle.com/software/index.html.

[8] Website. *Oracle Isql plus.*
http://www.oracleadvice.com/Tips/isqlplus.htm.

[9] Website. *Oracle-JDBC-Type-4-Treiber.*
http://technet.oracle.com/software/tech/java/sqlj_jdbc/software_index.htm.

[10] Website. *Oracle Thin-Treiber.*
http://otn.oracle.com/software/tech/java/sqlj_jdbc/htdocs/jdbc9201.html.

[11] Website. *orarun.rpm.*
ftp://ftp.suse.com/pub/suse/i386/supplementary/commercial/Oracle/orarun.rpm.

[12] Website. *Prometric Authorized Testing Centers.*
http://www.prometric.com.

[13] Website. *Tech on the Net, Data Types.*
http://www.techonthenet.com/oracle/datatypes.htm.

[14] Website. *Tech on the Net, Oracle Built-In Functions.*
http://techonthenet.com/oracle/functions/index.htm.

Index