







1 Oracle starten

1.1 Mit Hilfe der Betriebssystemdienste Oracle starten

- OracleServiceXE ... das eigentliche DBMS
- XETNSListener ... Programm zur Überwachung von Verbindungswünschen

| | | | |
|---|-----------------|-------------|----------------|
|  OracleJobSchedulerXE | | Deaktivi... | Lokales System |
|  OracleMTSRecoveryService | | Manuell | Lokales System |
|  OracleServiceXE | Wird ausgeführt | Manuell | Lokales System |
|  OracleXEClrAgent | | Manuell | Lokales System |
|  OracleXETNSListener | Wird ausgeführt | Manuell | Lokales System |
|  Orchestrator Service aktuali... | UsoSvc | Manuell | Lokales System |

Hinweis: es wird empfohlen diese Dienste“ manuell“ zu starten um den Startup Vorgang des Rechners nicht zu belasten. Der Start des Dienstes „OracleServiceXE“ nimmt (je nach HW Ausstattung) ca. 20 sec. In Anspruch.

1.2 In Sqlplus Oracle starten

Die Sqlplus - Kommandos

- Startup
- Shutdown

Starten bzw. stoppen das DBMS ebenfalls. Diese Kommandos werden im Kapitel über DBA Aufgaben noch näher beschrieben.

2 Sich bei Oracle anmelden

Hier wird der Weg beschrieben sich beim bereits gestarteten DBMS mit dem Programm “sqlplus” zu verbinden. Sqlplus bietet eine textbasierende Schnittstelle zu Oracle, man kann SQL Befehle sowie zusätzliche Kommandos eingeben und verarbeiten lassen. Technisch gesehen stellt Sqlplus einen “User Proces” dar der bei der Verbindung zu Oracle einen “Server Proces” generiert. In Windows Systemen ist der “Server Process” als Thread innerhalb von Oracle realisiert.

2.1 Sqlplus starten

```
C:\Users\tc>sqlplus

SQL*Plus: Release 11.2.0.2.0 Production on Di Apr 5 19:17:00 2016

Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter user-name:
```

Hinweis : Wichtig ist dass die Umgebungsvariable ORACLE_HOME auf das Server Verzeichnis des DBMS gesetzt wird. Dies sollte bei ordnungsgemäßer Installation von Oracle geschehen – kann aber auch nachträglich manuell durchgeführt werden :

```
C:\Users\tc>SET ORACLE_HOME=c:\oracle\app\oracle\product\11.2.0\server
```

2.2 Sich als Systemadministrator bei Oracle anmelden

Nachdem Sqlplus gestartet wurde kann man sich nun wie folgt anmelden :

```
Enter user-name: / as sysdba

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL>
```

Das >SQL Prompt ist der Hinweis dass die Anmeldung erfolgreich war, nun können SQL- und andere Kommandos eingegeben werden.

Hinweis: Diese Form der Anmeldung ist eigentlich ein „Not – Login“ für den Fall dass das DBMS nicht auf die Diskblöcke der User Accounts zugreifen kann, und nur demjenigen Betriebssystemuser vorbehalten der Oracle installiert hat !

3 Das ORACLE Data Dictionary

Jedes DBMS muss die Informationen über die tatsächlichen Nutzerdaten ebenso persistent speichern. Diese Metainformationen werden in Oracle als „Data Dictionary“ bezeichnet und ebenso in Form von Tabellen abgelegt. Als „sysdba“ User hat man Zugriff auf das gesamte Data Dictionary, hier einige wichtige Tabellen und Views :

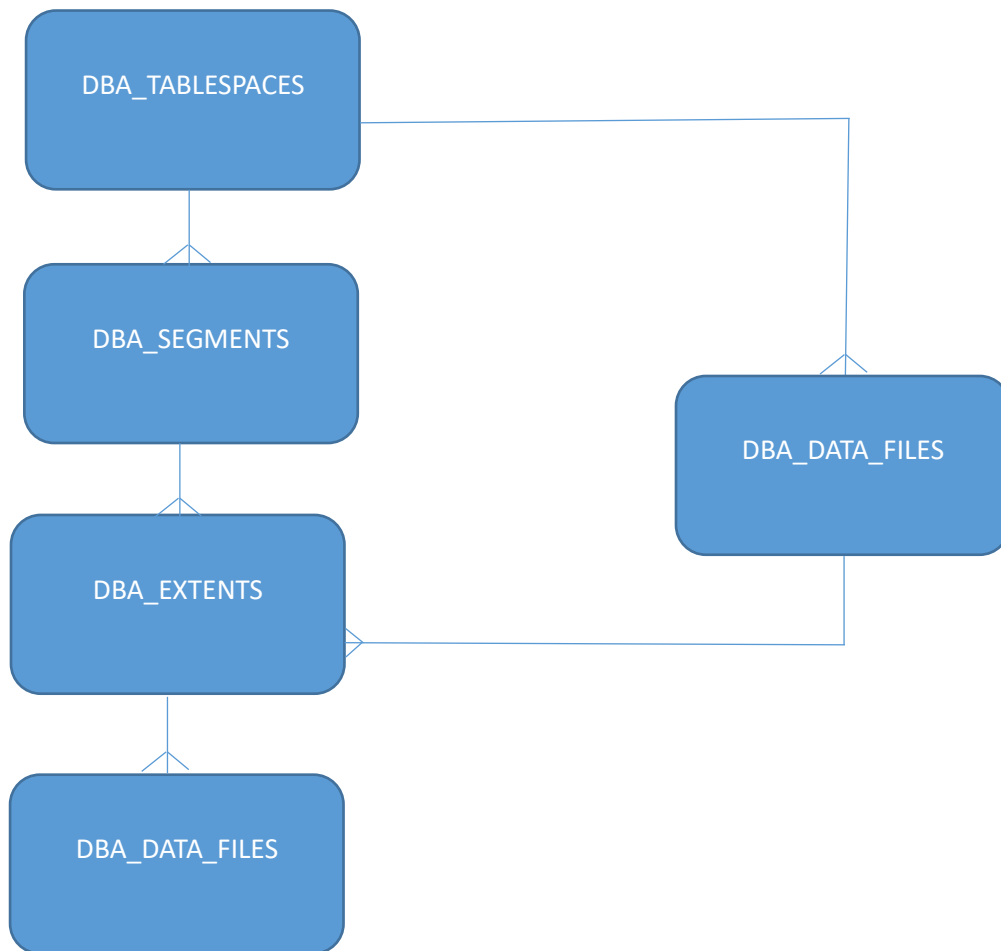
3.1 DBA_OBJECTS

DBA_OBJECTS enthält Informationen zu allen Datenbankobjekten und kann somit als allgemeine Informationsquelle angesehen werden.

- Wichtige Attribute
 - owner ... Name des Besitzers
 - object_name ... Name des DB Objekts
 - namespace ... Name des Namespaces (=Speicherverwaltung)
 - object_id ... eindeutige nr.
 - object_type ... Typ des DB Objekts (table, view, sequence,)

3.2 Speichermanagement in Oracle

Welches Datenbankobjekt wird in welcher physikalischen Datei gespeichert ? Diese Frage lässt sich in Oracle anhand des Data Dictionaries beantworten. Sie können alle Dateien der Oracle Datenbank anzeigen, die Größen und Anzahl der Blöcke erfahren und noch vieles mehr. Alle diese Informationen werden im DD (= Data Dictionary) in folgender Struktur abgelegt :



3.2.1 DBA_TABLESPACES

Tablespaces sind die „obersten“ Verwaltungseinheiten des sekundären Speichermanagements. Alle Daten einer logische Einheit (z.Bsp.: eine Firma) sollten in einem Tablespace verwaltet werden. Oracle bildet in einer Standardinstallation folgende wichtige Tablespaces aus :

- **USERS** ... Daten aller User
- **SYSTEM** ... Data Dictionary
- Ua.

Wichtige Attribute von **DBA_TABLESPACES** :

- **tablespace_name** ... Name des TS
- **block_size** ... Größe eines Oracle Blocks

- `max_size` ... maximale Größe des gesamten TS

3.2.2 DBA_SEGMENTS

Im Wesentlichen sind Segmente „Datenbankobjekte die Speicher brauchen“ also hauptsächlich Tabellen. Jeder Tablespace kann mehrere Segmente enthalten und jedes Segment gehört zu genau einem Tablespace. Jedes Segment besteht wiederum aus mehreren Extents !

Wichtige Attribute :

- `owner` ... Eigentümername
- `tablespace_name` ... zugehöriger Tablespace
- `segment_type` ... Typ des Segments (,TABLE', ...)
- `bytes` ... Anzahl der Bytes im Sekundärspeicher
- `blocks` ... Anzahl der Oracle Blöcke
- `extents` ... Anzahl der Extents
- `xxx_extents` ... Speicherverwaltung in Oracle

3.2.3 DBA_EXTENTS

Extents sind eine Einheit zusammengehörigen Speichers eines Segments. Jeder Extent wird in genau einem File gespeichert. Nachdem ein Segment aus mehreren Extents bestehen kann ist es grundsätzlich möglich dass ein Segment auf mehrere Dateien aufgeteilt wird !

Wichtige Attribute :

- `owner` ... Eigentümername
- `segment_name` ... zugehöriges Segment
- `segment_type` ... Typ des Segments (,TABLE', ...)
- `tablespace_name` ... zugehöriger Tablespace
- `extent_id` ... ID des Extents
- `bytes` ... Anzahl der Bytes im Sekundärspeicher
- `blocks` ... Anzahl der Oracle Blöcke
- `file_id` ... Fremdschlüssel auf „die Datei“

- `block_id` ... Fremdschlüssel auf den ersten Block

3.2.4 DBA_DATA_FILES

Jedes Datenfile in dem Blöcke von Oracledaten liegen erhält hier einen Eintrag.

Wichtige Attribute :

- `file_name` ... absoluter Dateiname
- `file_id` ... PK
- `bytes` ... allokierte Bytes

3.3 DBA_USERS

Jeder Oracle User erhält in dieser Tabelle einen Eintrag.

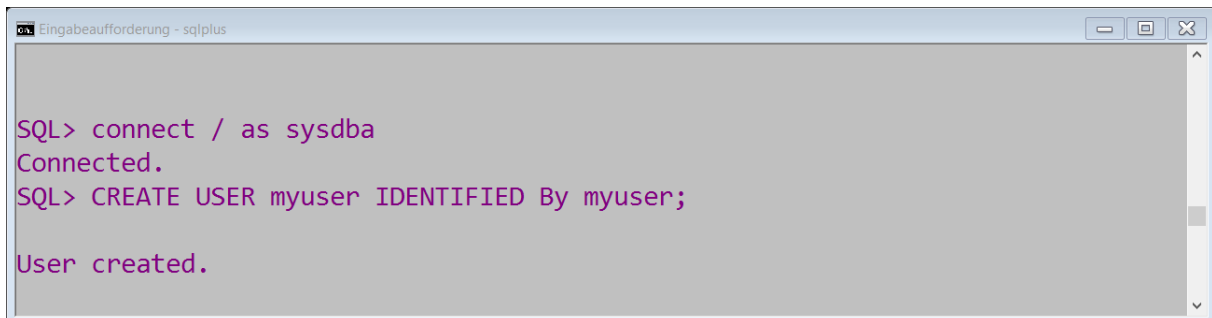
Wichtige Attribute :

- `username` ...
- `user_id` ... PK
- `password` ... wird verschlüsselt gespeichert !
- `account_status` ... LOCKED, UNLOCKED
- `default_tablespace` ... Tablespace für alle DB Objekte falls nicht explizit angegeben
- `temporary_tablespace` ... Tablespace für tmp. Daten

4 Benutzerverwaltung in Oracle

Nur ein User der das Privileg besitzt (CREATE USER) andere Benutzer anzulegen kann ein entsprechendes CREATE USER Statement absetzen.

4.1 Benutzer erzeugen



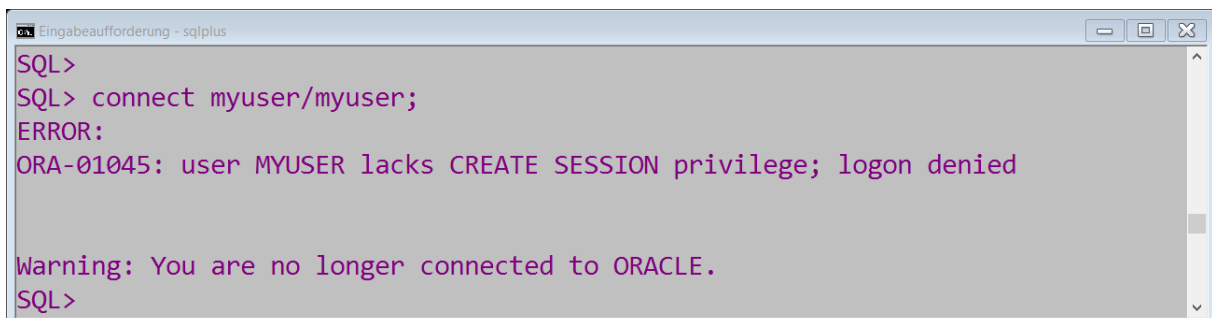
```
SQL> connect / as sysdba
Connected.
SQL> CREATE USER myuser IDENTIFIED By myuser;

User created.
```

Dies ist die einzige Stelle wo das Kennwort lesbar (hier: myuser) ist.

4.2 Benutzerprivilegien vergeben um eine Sitzung zu erzeugen

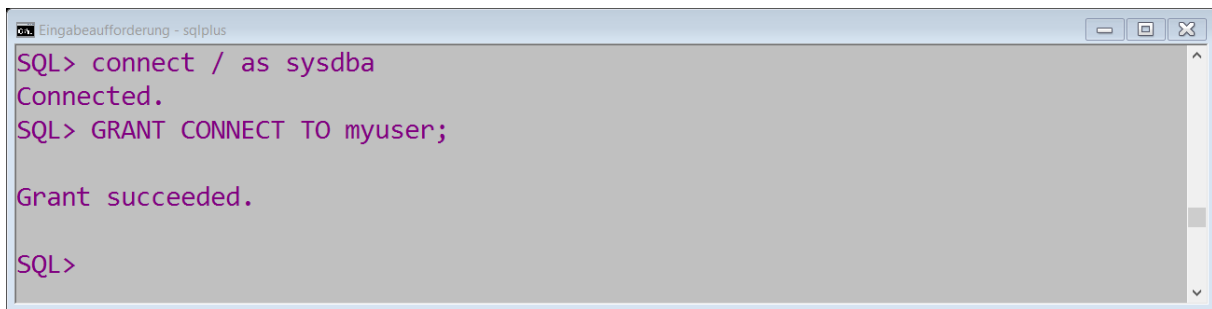
Um sich bei Oracle anzumelden muss der user aber noch mit entsprechenden Rechten ausgestattet werden. Noch besitzt der User ‚myuser‘ nicht das Recht eine Benutzersession zu eröffnen.



```
SQL>
SQL> connect myuser/myuser;
ERROR:
ORA-01045: user MYUSER lacks CREATE SESSION privilege; logon denied

Warning: You are no longer connected to ORACLE.
SQL>
```

Dieses Recht wird dem user ‚myuser‘ vom user ‚sysdba‘ erteilt (dieser hat das Privileg solche Privilegien zu vergeben). In Oracle gibt es die Möglichkeit mehrere Privilegien zu Rollen zusammenzufassen um die Verwaltung zu vereinfachen. So gibt es die Rolle ‚CONNECT‘ die einen User mit allen notwendigen Privilegien eine Sitzung zu eröffnen und Kommandos abzusetzen ausstattet.

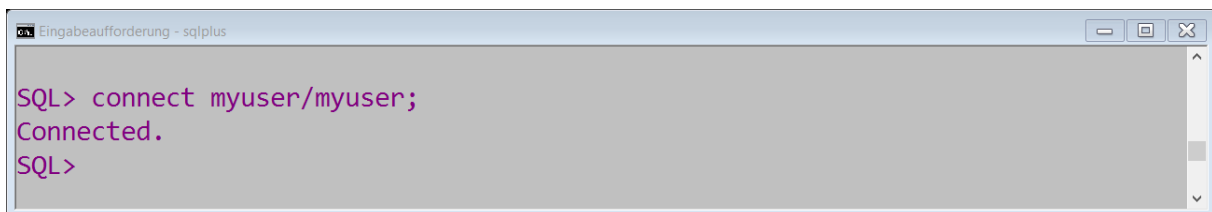


```
SQL> connect / as sysdba
Connected.
SQL> GRANT CONNECT TO myuser;

Grant succeeded.

SQL>
```

Nun sollte der Login für ‚myuser‘ möglich sein.



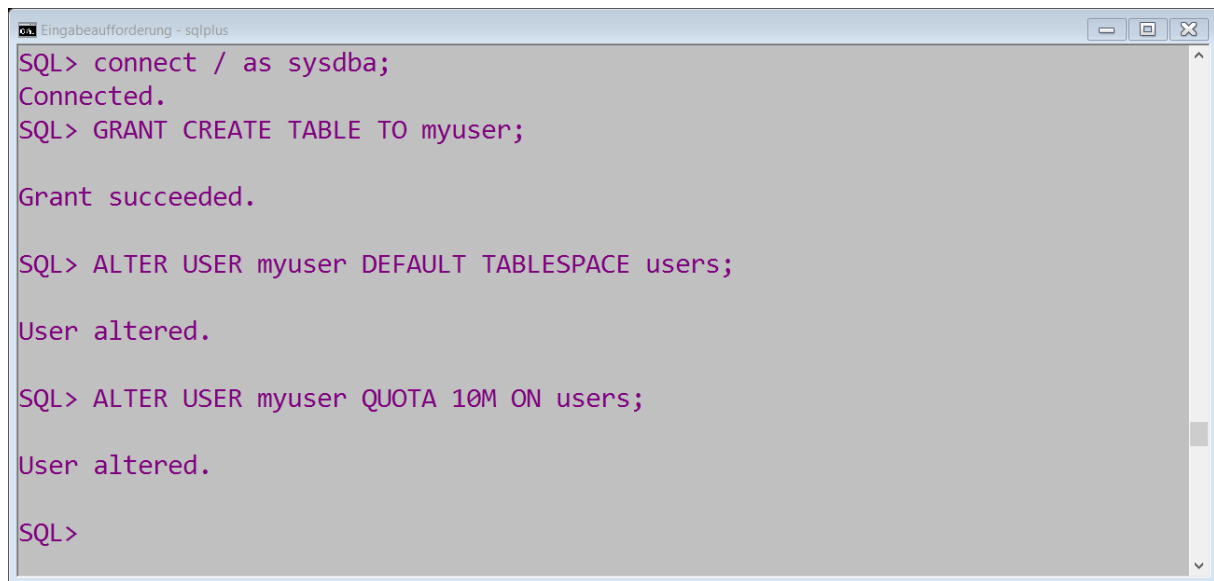
```
SQL> connect myuser/myuser;
Connected.
SQL>
```

4.3 Benutzerprivilegien vergeben um Tabellen anzulegen

Eine Sitzung mit Sqlplus zu erzeugen ist nur der Beginn, der nächste Schritt besteht darin Datenbankobjekte (Tabellen) zu erzeugen und zu bearbeiten. Dafür benötigt man wiederum spezielle Systemprivilegien und zwar :

- CREATE TABLE ... um Tabellen zu erzeugen und Datensätze zu verarbeiten
- Einen geeigneten Tablespace ... um die Datenätze in den „richtigen“ Dateien zu speichern
- QUOTA ... Angabe einer maximalen Speichermenge die jeder User in seinem Tablespace verbrauchen darf.

Die Vergabe der Rechte muss natürlich vom ‚sysdba‘ erfolgen .



```
SQL> connect / as sysdba;
Connected.
SQL> GRANT CREATE TABLE TO myuser;

Grant succeeded.

SQL> ALTER USER myuser DEFAULT TABLESPACE users;

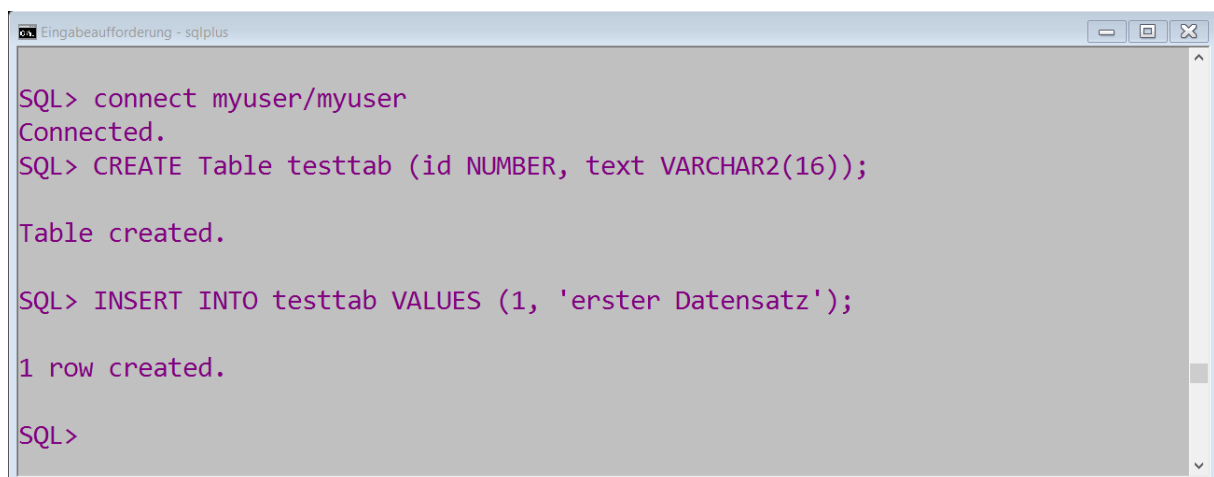
User altered.

SQL> ALTER USER myuser QUOTA 10M ON users;

User altered.

SQL>
```

Nun ist der User ‚myuser‘ bereit Tabellen zu erzeugen und Datensätze zu manipulieren.



```
SQL> connect myuser/myuser
Connected.
SQL> CREATE Table testtab (id NUMBER, text VARCHAR2(16));

Table created.

SQL> INSERT INTO testtab VALUES (1, 'erster Datensatz');

1 row created.

SQL>
```

5 Wichtige DBA Aufgaben

5.1 Starten und Stoppen von Oracle

Das Stoppen und Starten von Oracle kann auch innerhalb von Sqlplus erfolgen, Sinn der Sache ist das DBMS kontrolliert in einen definierten Startup Zustand zu bringen um diverse Wartungsarbeiten durchzuführen. Oracle kennt folgende Startup Zustände :

SHUTDOWN

Die Datenbank Instanz ist gestoppt und die Datenbank geschlossen. D.h. es laufen keine Oracle Prozesse, es ist kein Hauptspeicher allokiert und alle Datenbankdateien sind geschlossen.

NOMOUNT

Im Zustand NOMOUNT ist die Datenbank Instanz gestartet. D.h. die Prozesse (Windows: Threads) laufen und der Hauptspeicher (SGA) ist allokiert. Die Initialisierungsdatei ("initora") wurde gelesen, so dass diverse Informationen über die Datenbank bereits bekannt sind (z.B. wo sich die controlfiles) befinden.

Dieser Zustand wird normalerweise nicht manuell herbeigeführt. Lediglich für die Befehle create database, create controlfile , sowie für das Einschalten des Archivelog-Modus mit dem Befehl archive log start wird er benötigt.

MOUNT

In der Initialisierungsdatei sind die control files mit ihren Pfaden eingetragen. Mehrere Einträge bedeuten, dass die Dateien gespiegelt sind. Sie werden gelesen und abgeglichen. Sollten Differenzen zwischen den Dateien auftauchen erscheint eine Fehlermeldung und die Datenbank kann nicht gestartet werden. Der Administrator muss dann manuell durch kopieren dafür sorgen, dass alle Dateien identisch sind. Er sollte dazu die neueste Datei verwenden!

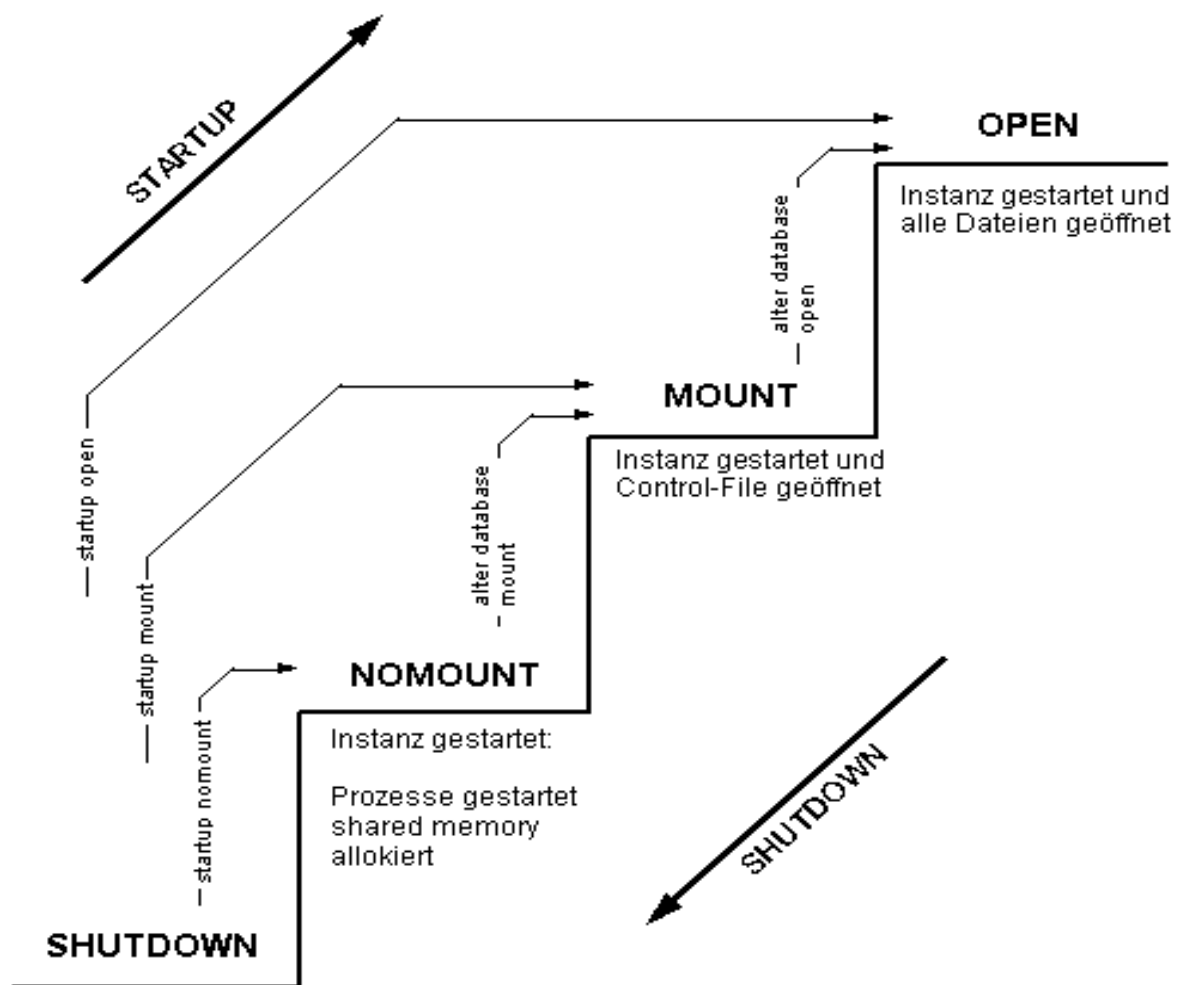
In den Control-Dateien sind Informationen über die eigentlichen Datenbankdateien abgelegt. Die Dateien sind aber noch nicht geöffnet, so dass in diesem Zustand einigen administrative Aufgaben durchgeführt werden. Typisches Beispiel ist das Umbenennen bzw. verschieben von Dateien:

➤ `ALTER DATABASE RENAME FILE '<old_file>' to '<new_file>';`

Achtung! Es handelt sich nur um die Bekanntmachung eines neuen Namens. Deshalb ist vorher die Datei auf Betriebssystemebene umzubenennen bzw. zu kopieren.

OPEN

Jetzt werden alle Datenbankdateien inkl. Redo-Log-Dateien geöffnet und deren Konsistenz geprüft. Wenn die Datendateien nicht auf dem aktuellen Stand sind, weil z.B. zuletzt mit shutdown abort beendet wurde, so wird versucht mit Hilfe der Online-Redo-Log-Dateien den aktuellen Zustand wieder herzustellen. Sollte dieses nicht gelingen, so ist ein Media Recovery und damit manuelles Eingreifen notwendig.



Wie in der Grafik ersichtlich können die einzelnen Zustände durch folgende Kommandos erreicht werden :

- `> shutdown` ... wie shutdown NORMAL
- `> shutdown normal` ... Zustand SHUTDOWN erst nachdem alle Sitzungen beendet werden
- `> shutdown TRANSACTIONAL` ... Zustand SHUTDOWN erst nachdem alle Transaktionen beendet werden
- `> shutdown IMMEDIATE` ... Zustand SHUTDOWN nachdem alle Transaktionen ordnungsgemäß abgebrochen werden
- `> shutdown ABORT` ... Zustand SHUTDOWN sofort, hinterlässt inkonsistente Dateien, beim nächsten startup muss Oracle diese erst „reparieren“
- `> startup nomount` ... führt zum Zustand NOMOUNT, nur von SHUTDOWN möglich
- `> startup mount` ... führt zum Zustand MOUNT, nur von SHUTDOWN möglich

- > startup open ... führt zum Zustand OPEN, nur von SHUTDOWN möglich
- > startup ... wie startup open

Alternativ kann von einem „niederen“ Zustand in einen „höheren“ Zustand mit dem Kommando :

- ALTER DATABASE MOUNT ... nach MOUNT „hinauf“ wechseln
- ALTER DATABASE OPEN ... nach „OPEN“ hinauf wechseln

Bemerkung: Der Datenbankzustand kann niemals schrittweise hinuntergefahren werden – es geht nur mit „shutdown“ ganz hinunter und dann evtl. schrittweise hinauf.

5.2 Oracle Initialisierungsdateien

Beim Starten von Oracle werden aktuelle Systemparameter aus einer Initialisierungsdatei gelesen. Diese wird bei der Installation von Oracle bereitgestellt und liegt als Binärdatei (bitte nicht manuell verändern !!!) per Default in folgendem Verzeichnis :

ORACLE_HOME\database\initORACLE_SID.ora

(ORACLE_SID = Name ihrer Oracle Installation, z.Bsp.: ‚XE‘)

In einer Std. XE Installation also :

| C (C:) > oraclexe > app > oracle > product > 11.2.0 > server > database | | | |
|---|------------------|--------------|-------|
| Name | Änderungsdatum | Typ | Größe |
| hc_xe.dat | 21.03.2015 01:48 | DAT-Datei | 2 KB |
| initXE | 21.03.2015 01:00 | ORA-Datei | 1 KB |
| oradba | 29.05.2014 13:05 | Anwendung | 31 KB |
| oradim | 05.04.2016 19:21 | Textdokument | 9 KB |
| PWDXE | 21.03.2015 01:00 | ORA-Datei | 2 KB |

Die Default Init Datei wird auch als SPFILE bezeichnet und diese Datei kann mit dem Befehl:

```
SQL> ALTER SYSTEM SET ....
```

verändert werden sodass die Änderungen auch beim nächsten startup wirksam sind.

Alternativ kann man einen Oracle startup Vorgang auch mit einer eigenen Textparameterdatei durchführen. Diese kann dann sehr wohl mit einem Texteditor geändert werden. Allerdings hat so ein startup Vorgang nur einmalige Auswirkung – beim nächsten mal wird wieder das Default init File verwendet.

Um aus einem vorhandenen SPFILE eine Text Initialisierungsdatei zu erzeugen (die man dann später ändern kann) gibt es den Befehl :

```
SQL> ALTER SYSTEM CREATE PFILE = '<Pfad zum PFile>' FROM SPFILE = '<Pfad zum SPFile >';
```

bzw. umgekehrt :

```
SQL> ALTER SYSTEM CREATE SPFILE = '<Pfad zum SPFile>' FROM PFILE = '<Pfad zum PFile>';
```

Die Pfadangaben (inkl. =) obiger Befehle sind optional – Oracle verwendet dann Defaultverzeichnisse und Defaultnamen.

5.3 Controlfile(s) in Oracle

Beim Anlegen einer Oracle Datenbank wird ein einziges Control File angelegt. Im Control File werden alle Informationen über den physikalischen Aufbau der Oracle Datenbank wie Pfaden und Dateinamen (der Datafiles und Redologs), Konsistenzinformationen, Informationen zur Datenbanksicherung mit RMAN gespeichert.

Der Speicherort des Control Files ist aber frei konfigurierbar. Nach der Empfehlung von ORACLE wird der Name des Control File mit .ctl erweitert.

Wenn das Control File defekt oder nicht vorhanden ist, dann kann die nicht geöffnet werden. Deshalb ist sinnvoll das Control File auf mehreren Festplatten zu spiegeln und mit Oracle Mitteln regelmäßig zu sichern. In der MOUNT Phase werden die gespiegelten Control Files geöffnet und miteinander verglichen. Sind alle Control Files absolut identisch, dann werden die Control Files geöffnet und die MOUNT Phase erfolgreich abgeschlossen. Somit sind alle Verweise auf die Datenbankdateien mit den Konsistenzinformationen bekannt.

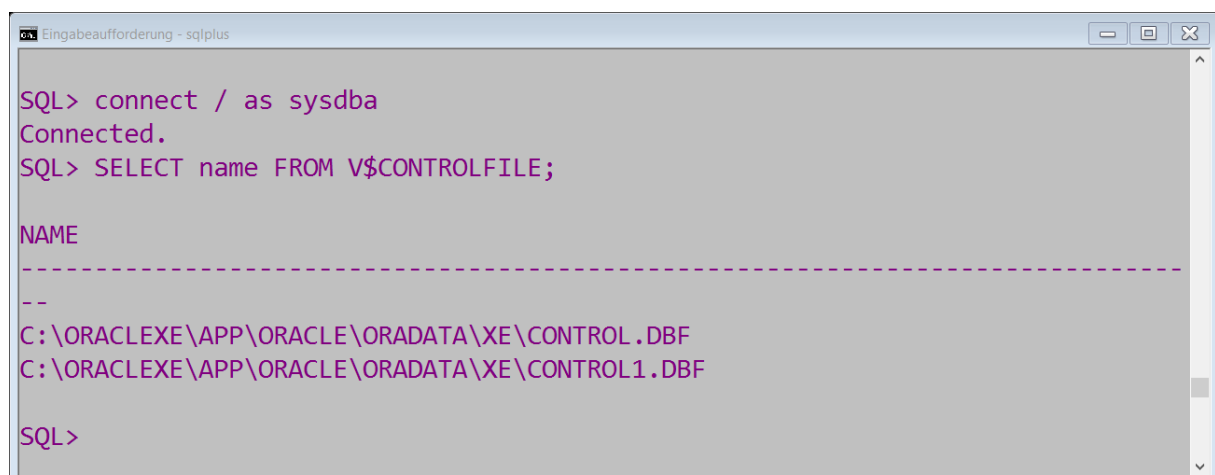
Werden in MOUNT Phase z.B. Datafiles oder Redologs hinzugefügt, umbenannt, verschoben oder gelöscht, dann werden diese Änderungen in die geöffneten Control Files angepasst. Auch die Archivierungsmodus oder Flashback-Modus der Datenbank kann nur in der MOUNT Phase aktiviert oder deaktiviert werden. Auch die Ausführung eines Datenbank-Flashback – das Zurückspielen der DB auf einen älteren Zeitpunkt kann nur in der MOUNT Phase umgesetzt werden. In der MOUNT Phase können z.B. Datenbankdateien verschlossen und umbenannt werden.

Ein Control File enthält folgende Informationen:

- Name und Timestamp (Erstellungszeitpunkt) der Datenbank
- Pfade und Namen aller zur Datenbank gehörenden Datafiles und Redologs
- Status-Informationen aller verzeichneten Tablespaces sowie der Datafiles und Redologs
- Die aktuelle Log Sequence Number der Redologs (SCN)
- Informationen zur Redologs-Historie
- Informationen zur Archivierung der Redologs
- Informationen zu Datenbanksicherungen mit RMAN
- Konsistenz-Informationen der Datenbank

5.3.1 Wo sind die Controlfiles und welche gibt es

Mit der Data Dictionary View V\$CONTROLFILE ermittelt man Pfad und Dateinamen aller Controlfiles.



```
Eingabeaufforderung - sqlplus

SQL> connect / as sysdba
Connected.
SQL> SELECT name FROM V$CONTROLFILE;

NAME
-----
--
C:\ORACLEXE\APP\ORACLE\ORADATA\XE\CONTROL.DBF
C:\ORACLEXE\APP\ORACLE\ORADATA\XE\CONTROL1.DBF

SQL>
```

5.3.2 Anpassen des Serverparameters CONTROL_FILES

Vor jeder strukturellen Änderung innerhalb des Control Files sollte das aktuelle Control File gesichert werden und anschließend wird der Parameter CONTROL_FILES um das aktuelle Control File (Pfad und Name) erweitert werden.

Unter struktureller Änderung ist z.B. wenn eine Datendatei hinzugefügt oder entfernt wird, ein Tablespace online oder offline gesetzt wird, ein Tablespace in den schreibgeschützten oder in den schreibbaren Modus versetzt wird.

```
SQL> create pfile = '/db_backup/controlfile_backup.ctl' from spfile;
```

oder Control File im laufenden Betrieb durch eine Kopie sichern (binäre Datei)

```
SQL> alter database backup controlfile to '/db_backup/controlfile_backup.ctl';
```

oder Control File mit einem Trace Dumpen (Textdatei)

```
SQL> alter database backup controlfile to trace;
```

Hier wird die Trace-Datei bis Oracle 10g im Ordner gespeichert, auf den der Parameter USER_DUMP_DEST zeigt. Ab Oracle 11g wird die Trace-Datei im Ordner ADR_HOME/trace abgelegt.

Nun kann man die Parameter des SPFiles interaktiv ändern :

```
SQL> alter system set control_files =
```

```
    '/u01/app/oracle/oradata/myorcl/control1.ctl'
```

```
    ,'/u02/app/oracle/oradata/myorcl/control2.ctl'
```

```
    scope = spfile;
```

Hinweis: scope = spfile gibt an dass das Default SPFile zu ändern ist.

5.3.3 Herunterfahren der Instanz

Um die Änderungen anzuwenden muss das Controlfile kopiert werden – dies geht nur wenn die Oracle Instanz heruntergefahren ist !

```
SQL> shutdown immediate;
```

5.3.4 Das Kopieren des Control Files in das neue Verzeichnis

Nun kann im Betriebssystem das Controlfile kopiert werden :

```
>cp /u01/app/oracle/oradata/myorcl/control1.ctl  
/u02/app/oracle/oradata/myorcl/control2.ctl
```

5.3.5 Starten der Instanz und Überprüfung der neuen Orte der Kontrolldateien

```
SQL> startup
```

```
SQL> select status, name from V$CONTROLFILE;
```

```
SQL> show parameter control_files;
```

```
SQL> select name, status from V$CONTROLFILE;
```

```
SQL> select name, value from V$PARAMETER where name =  
        'control_files';
```


5.4 Logfiles verwalten

RedoLog Dateien oder kurz Logfiles sind Dateien welche die letzten Änderungsstatements (INSERT, UPDATE) beinhalten. Die Datensätze werden nach jedem Checkpoint bzw. Commit von der Instanz in die Logfiles geschrieben. Der Oracle Thread ‚DatabaseWriter‘ arbeitet die Logfiles im Hintergrund ab und

5.4.1 Was sind Redo - Log Dateien ?

Die wichtigste Einheit um Transaktionen schnell zu verarbeiten und um Aufräumarbeiten bei Systemabstürzen und sonstigen Fehlersituationen (= Recovery) durchzuführen sind RedoLog Dateien. Diese bestehen aus zumindest 2 Dateien welche alle Änderungen des Datenbestands (INSERT, UPDATE, DELETE) speichern.

5.4.2 Inhalt der Redo Log Dateien

RedoLog Dateien werden mit bestimmten Datensätzen gefüllt – sogenannte RedoLog Entries. Jeder Entry besteht wieder aus einer Menge von Änderungsbeschreibungen (= Changevector), wobei jede Änderungsbeschreibung die Information enthält welcher Diskblock genau welche Änderung erfährt. Wenn man also z.Bsp. von einem Mitarbeiter das Gehalt ändert erhält man einen Redo Record mit den Änderungen für den data segment block der Tabelle, den undo segment block und den transaction table des undo segments.

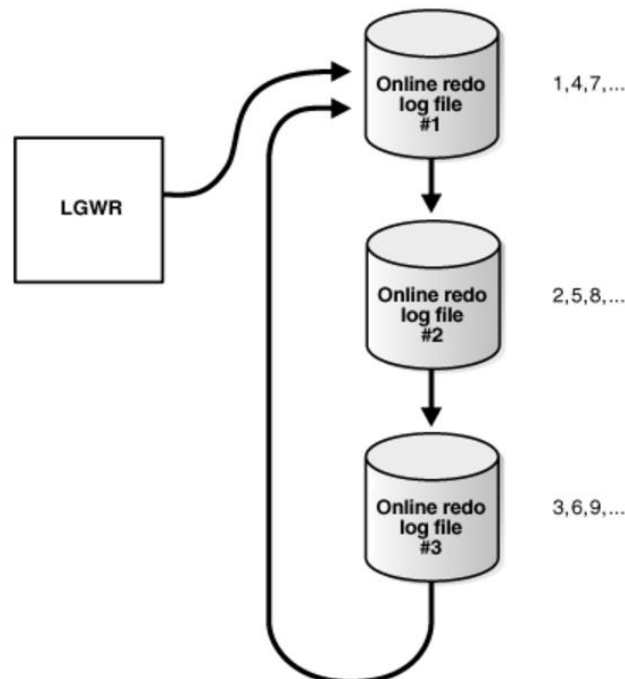
Somit kann man mit diesen Informationen aus einem alten Datenbestand alle Änderungen nachführen und den neuen Datenbestand herleiten.

Redo Records werden innerhalb der Instanz im sogenannten RedoLog Buffer zwischengespeichert und periodisch in eine der RedoLog Dateien geschrieben. Dies geschieht durch einen Hintergrundprozess (oder in Windowssystemen in einem Thread) dem Log Writer (LGWR). Immer wenn eine Transaktion einen COMMIT absetzt schreibt der LGWR alle Redo Records der Transaktion vom SGA Puffer in die gerade aktive RedoLog Datei und vergibt eine SystemChangeNumber (SCN) um diese Redo Records zu kennzeichnen (funktioniert wie ein Fremdschlüssel auf die Transaktion). Nur wenn dieser Vorgang abgeschlossen ist und alle Redo Records gespeichert sind wird der User Prozess (= das Programm welches der Anwender bedient) informiert dass der COMMIT erfolgreich war. Dieser löst dann eine entsprechende Meldung am Schirm des Users aus.

Redo Records könne auch schon bevor die Transaktion endet in die Redo Dateien gelangen. Dies geschieht falls die gerade aktive RedoLog Datei voll ist und ein sogenannter LogSwitch erfolgt oder eine andere Transaktion einen COMMIT absetzt. Falls notwendig kann das DBMS diese Änderungen wieder rückgängig machen (= Rollback).

5.4.3 Wie erfolgt das Schreiben in die RedoLog Dateien

Der RedoLog einer Datenbank besteht aus mindestens 2 RedoLog Dateien. Die Datenbank verlangt mindestens 2 Dateien um sicherzustellen dass eine Datei verfügbar ist um Redo Records zu



beschreiben während die andere noch bearbeitet und ggf. archiviert wird – falls das DBMS im ARCHIVELOG Modus betrieben wird.

LGWR beschreibt die RedoLog Dateien in einer zyklischen Reihenfolge. Zuerst die Erste, dann die Zweite usw. und am Ende wieder die Erste.

Beschriebene RedoLog-Dateien sind für den LGWR wieder verfügbar abhängig vom Archivierungszustand des DBMS. Nachdem eine RedoLog-Datei befüllt wurde müssen die Änderungen noch von einem weiteren Hintergrundprozess, dem Database Writer (DBWR) in die entsprechenden Disk-Blöcke übertragen werden. Falls die Archivierung eingeschaltet ist muss die Datei auch noch kopiert werden. Erst wenn diese Vorgänge abgeschlossen sind kann der LGWR diese Datei wieder verwenden.

Diejenige Datei, welche gerade vom LGWR verwendet wird, wird als „**current**“ RedoLog-Datei bezeichnet. Jene Dateien, welche noch nicht fertig abgearbeitet wurden und für ein eventuelles Instance Recovery (nach Systemabstürzen) verwendet werden werden als „**active**“ bezeichnet. Wenn dies nicht mehr der Fall ist erhält sie den Zustand „**inactive**“.

Falls die Archivierung eingeschaltet wurde kann das DBMS eine „active“ RedoLog Datei nicht verwenden – falls dies nicht der Fall ist wird die erste „active“ Datei beim nächsten LogSwitch verwendet.

5.4.4 Log Guppen und Log Member

Aus Sicherheitsgründen ist es ratsam jede RedoLog Datei zu spiegeln. Die Menge aller gespiegelten RedoLog Dateien nennt man auch eine „**Log Group**“ bestehend aus mehreren „**Log Membern**“. Es ist eine wesentliche Aufgabe des DBA die Gruppen einzurichten (siehe unten).

5.4.5 Log Switches und Log Sequence Numbers

Ein Log Switch ist der Zeitpunkt an dem der LGWR eine neue RedoLog Datei zu beschreiben beginnt. Dies geschieht üblicherweise wenn die aktuelle RedoLog Datei voll ist – kann aber auch künstlich herbeigeführt werden.

Oracle weist jeder RedoLog Datei eine neue Log Sequence Number (= LSN) nachdem der Log Switch erfolgte. Diese LSN wird in der Datei gespeichert.

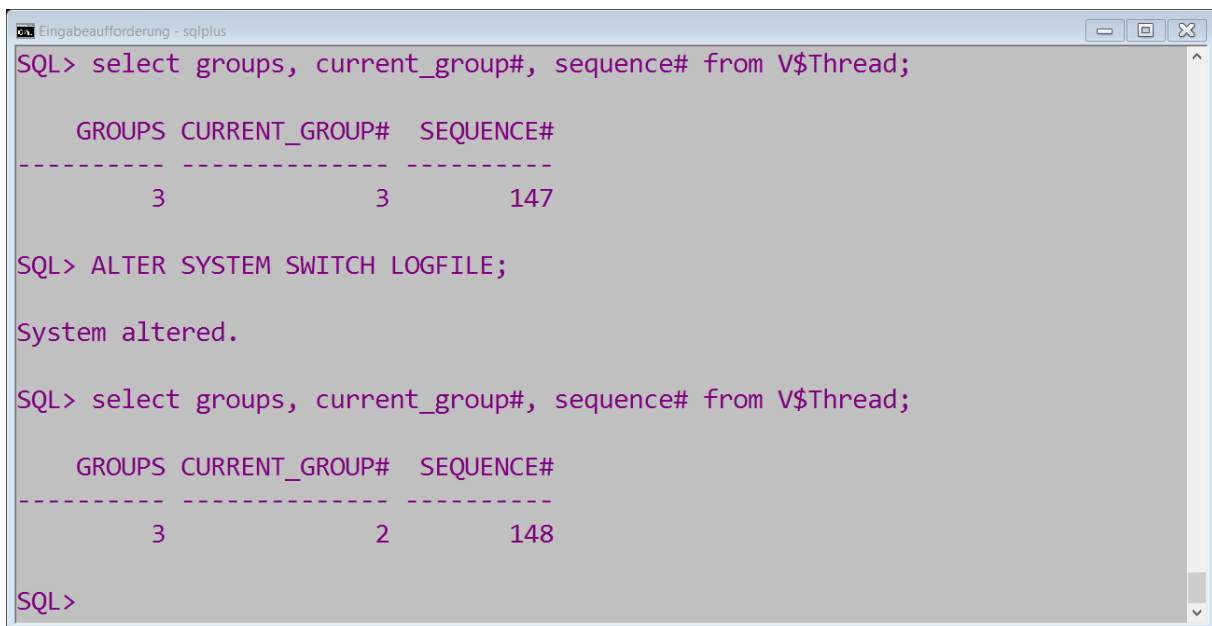
Anhand der LSN kann die Reihenfolge der RedoLog Dateien festgestellt werden.

5.4.6 Wichtige Views

- V\$LOG ... pro Log Gruppe ein Eintrag
 - GROUP# ... Gruppen Nr.
 - MEMBERS ... Anzahl der Log Member der Gruppe
 - STATUS --- CURRENT / ACTIVE / INACTIVE
 - SEQUENCE# ... SCN der Log Datei
 - BYTES ... Größe einer Datei
- V\$LOGFILE ... pro Log Member ein Eintrag
 - GROUP# ... Gruppen Nr. der RedoLog Datei
 - STATUS ... Status der RedoLog Datei
 - MEMBER ... absoluter Pfad des Log Members.

- V\$THREAD ... Info über LGWR
 - GROUPS ... Anzahl der möglichen LogGroups
 - CURRENT_GROUP# ... welche Log Group ist gerade aktiv
 - SEQUENCE# ... aktuelle SCN

Die Auswirkungen des Befehls: „ALTER SYSTEM SWITCH LOGFILE“ können in V\$THREAD gesehen werden.



```
SQL> select groups, current_group#, sequence# from V$Thread;

  GROUPS CURRENT_GROUP# SEQUENCE#
-----
      3             3      147

SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

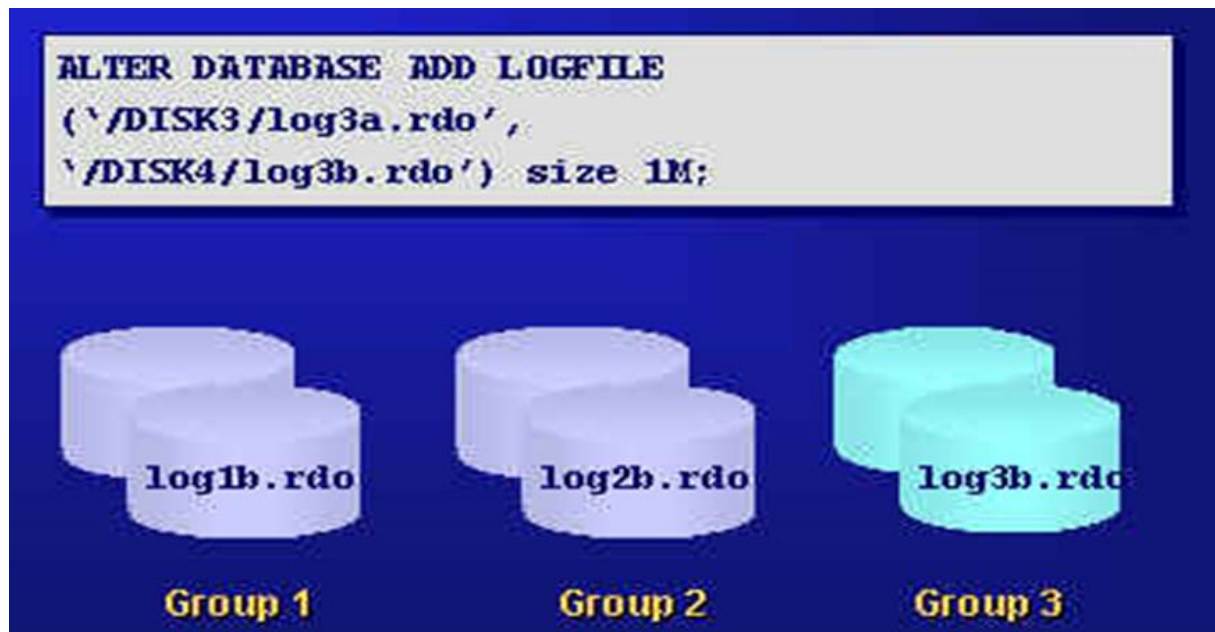
SQL> select groups, current_group#, sequence# from V$Thread;

  GROUPS CURRENT_GROUP# SEQUENCE#
-----
      3             2      148

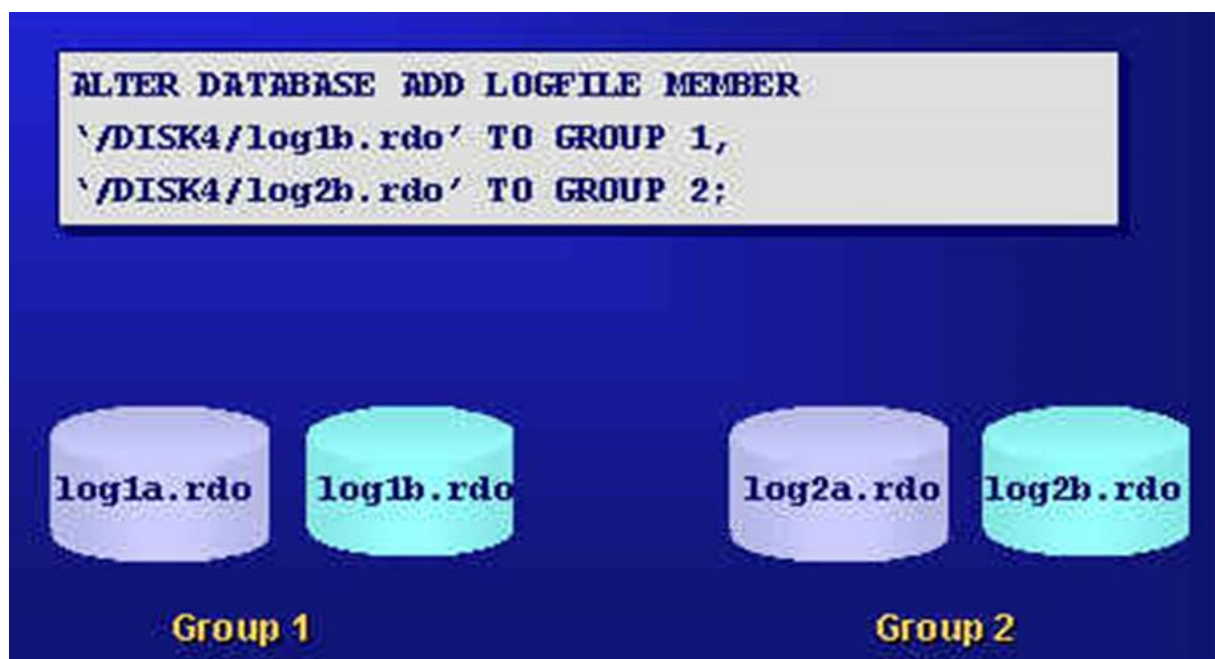
SQL>
```

5.4.7 Wichtige Kommandos in SQLPlus

5.4.7.1 Log Gruppe hinzufügen



5.4.7.2 Log Member hinzufügen



5.4.7.3 Log Member entfernen

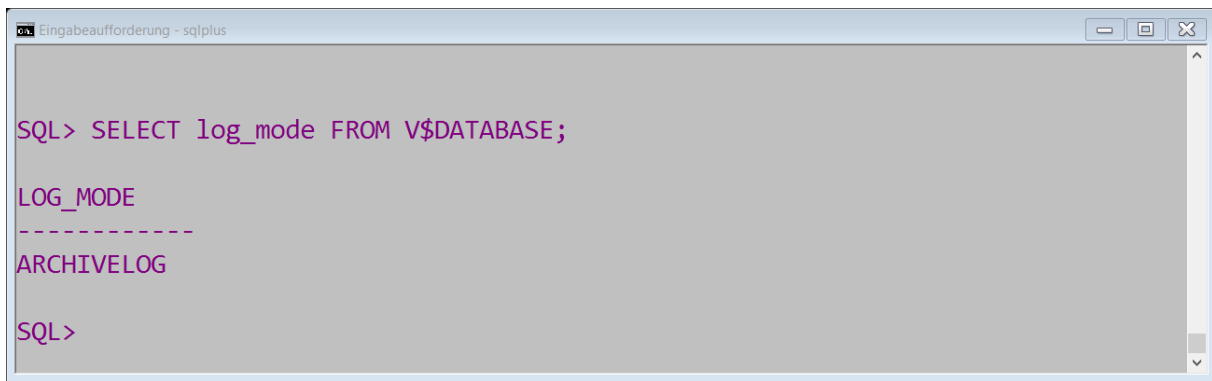


5.4.7.4 Log Gruppe Entfernen



5.4.7.5 Archivierungsmodus feststellen

SQL > SELECT log_mode FROM V\$DATABASE;



```
SQL> SELECT log_mode FROM V$DATABASE;

LOG_MODE
-----
ARCHIVELOG

SQL>
```

5.4.7.6 Datenbank in den Archivierungsmodus setzen

- DB in den Modus “mount” versetzen

SQL> shutdown immediate;

SQL> startup

- Modus umstellen

SQL> ALTER DATABASE ARCHIVELOG;

- DB öffnen

SQL> ALTER DATABASE OPEN;

- Automatische Archivierung aktivieren

SQL> ALTER SYSTEM ARCHIVE LOG START;

- automatische Archivierung für jedenStart einer Instanz

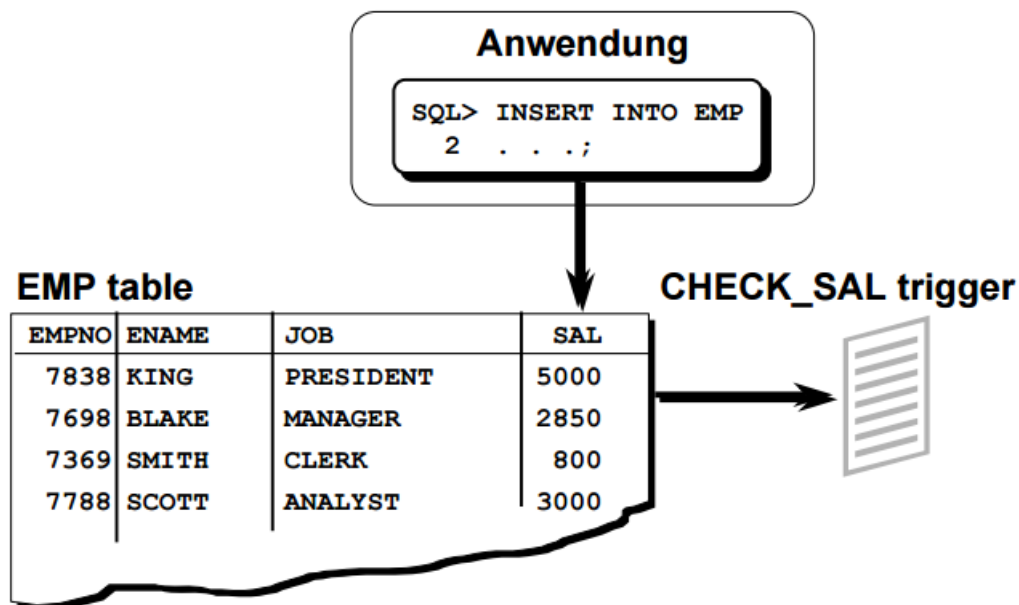
SQL> ALTER SYSTEM log_archive_start=true SCOPE=SPFILE;

6 Trigger in Oracle

6.1 Überblick über Trigger

Ein Trigger ist ein PL/SQL Block, der implizit aufgerufen wird, wenn ein bestimmtes Ereignis eintritt.
Ein Trigger kann ein Datenbanktrigger oder ein Anwendungstrigger sein.

Bespiel: ein INSERT Statement löst einen Trigger aus welcher das Gehalt eines Mitarbeiters überprüft.



6.2 Aufbau und Bestandteile eines Triggers

- Trigger timing: BEFORE oder AFTER
- Triggering event: INSERT oder UPDATE oder DELETE
- Tabellename: On <table>
- Trigger Typ: Row oder Statement
- WHEN Klausel: Einschränkungsbedingung
- Trigger body: DECLARE
....
BEGIN
...
END;

6.2.1 Trigger Timing

Es geht um die Frage „Wann soll der Trigger feuern?“

- BEFORE: Der Code im Trigger-Body soll ausgeführt werden bevor das DML Ereignis ausgeführt wurde.
- AFTER: Der Code im Trigger-Body soll ausgeführt werden nachdem das DML Ereignis ausgeführt wurde.
- INSTEAD OF: Der Code im Trigger-Body soll an Stelle des Ereignisses ausgeführt werden. Wird für VIEWS verwendet, die nicht anders modifiziert werden können.

6.2.2 Triggering Event

Hier geht es um die Frage: „Welche DML Operation soll die Ausführung des Triggers verursachen?“

- INSERT
- UPDATE
- DELETE
- oder besondere DB Events (login, ...)

6.2.3 Trigger Typ

Hier geht es um die Frage: „Wie oft soll der Trigger ausgeführt werden, wenn das Triggerereignis eintritt?“

- Statement: Der Trigger wird nur einmal für das Ereignis ausgeführt (Standard).
- Row: Der Trigger wird für jedes Tupel ausgeführt, bei dem das Ereignis auftritt.

6.2.4 Trigger Body

Hier geht es um die Frage: „Welche Aktion soll der Trigger ausführen?“. Dies ist der eigentliche Hauptteil des Triggers. Der Trigger Body ist definiert wie ein anonymer PL/SQL Block.

| | |
|-------------|--|
| [DECLARE] | ... optional, Deklaration von Variablen |
| BEGIN | ... Schlüsselwort, Beginn der Statements |
| [EXCEPTION] | ... optional, Auffangen von Laufzeitausnahmen |
| END; | ... schlüsselwort, Ende des Triggers (inkl. „;“) |

6.2.5 Ausführungsreihenfolge falls ein Statement mehrere Datenätze ändert und mehrere Trigger definiert wurden.

Es ist möglich dass auf ein Ereignis gleichzeitig sowohl anweisungsbezogene als auch zeilenbezogene Trigger existieren. Dann werden diese in folgender Reihenfolge abgearbeitet :

1. BEFORE Statement Trigger
2. BEFORE Row Trigger für jeden betroffenen Datensatz
3. AFTER Row Trigger für jeden betroffenen Datensatz
4. AFTER Statement Trigger

6.2.6 Trigger Beispiel: „secure_emp“

```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON emp
BEGIN
    IF (TO_CHAR (sysdate,'DY') IN ('SAT','SUN')) 5 OR
        (TO_CHAR(sysdate,'HH24') NOT BETWEEN '08' AND '18' THEN
        RAISE_APPLICATION_ERROR (-20500,
            'You may only insert into EMP during normal 9 hours. ');
    END IF;
END;
```

Erklärung :

- Der Trigger wird vor jedem INSERT Statement in die Tabelle ,emp‘ ausgeführt
- Das IF Statement prüft ob der aktuelle Wochentag auf Samstag oder Sonntag fällt oder die Uhrzeit außerhalb der Arbeitszeit liegt
- Falls ja: RAISE_APPLICATION_ERROR löst einen Systemfehler aus und das Statement wird nicht durchgeführt !

6.2.7 Verwendung von :new und :old

Mit Hilfe der Identifier :old und :new kann im Trigger auf die aktuellen Daten zugreifen. Beide sind Bindevariablen vom Typ : trigger_table%ROWTYPE – also der selbe Datentyp wie die zugrundeliegende Tabelle. Eine Referenz wie :new.attribut ist gültig, wenn attribut ein Attribut der triggernden Tabelle ist. :new und :old sind Pseudodatensätze.

Bedeutung von :new und :old

| Triggernde Anweisung | :old | :new |
|----------------------|--------------------------------------|--------------------------------------|
| INSERT | Undefiniert | Werte des neuen Datensatzes |
| UPDATE | Werte des alten Datensatzes | Werte des aktualisierten Datensatzes |
| DELETE | Werte des ursprünglichen Datensatzes | Undefiniert |

6.2.8 Trigger Beispiel: „derive_comission_pct“

```
CREATE OR REPLACE TRIGGER derive_commission_pct
BEFORE INSERT OR UPDATE OF sal ON emp
FOR EACH ROW
WHEN (new.job = 'SALESMAN')
BEGIN
    IF INSERTING THEN
        :new.comm := 0;
    ELSE
        IF :old.comm IS NULL THEN
            :new.comm := 0;
        ELSE
            :new.comm := :old.comm * (:new.sal/:old.sal);
        END IF;
    END IF;
END;
```

Erklärung :

- Der Trigger feuert vor INSERT oder UPDATE der Spalte ‚sal‘ der emp Tabelle
- Bei INSERT wird das Attribut ‚comm‘ auf 0 gesetzt
- Bei UPDATE wird ‚comm‘ auf 0 gesetzt falls bisher ‚comm‘ auf 0 war, sonst wird dieses Attribut um das Verhältnis der Gehaltsentwicklung vergrößert.
- Es genügt :new.comm zu setzten da das nachfolgende INSERT dann diesen Wert einfügt !

