Kapitel 3

Administration von Datenbanksystemen

Wer sich heute auf einer modernen Linux-Distribution ohne weiteres Dazutun eine MySQL-Datenbank installieren will, bekommt bereits in über der Hälfte der Fälle Maria DB. Das ist ja auch nicht weiter schlimm, soll Maria DB doch ein Drop-in-Replacement für MySQL sein.

3.1 Geschichtliche Entwicklung von MySQL bzw. MariaDB

Im Jahre 2008 wurde MySQL von Sun Microsystems gekauft – nur ein Jahr später wurde Sun von Oracle übernommen. Dies wiederum passte dem Gründer von MySQL, Monty Widenius, nicht und er gründete eine neue Firma namens Maria DB, in der er mit ein paar Ex-MySQL-Entwicklern einen Branch von MySQL weiterpflegt. Das geschah zu Zeiten von Maria DB 5.5, und zu diesem Zeitpunkt konnte man noch getrost von einem Drop-in-Replacement sprechen.
Aber die Zeit ging ins Land, neue Features kamen hinzu. Maria DB und MySQL fingen an, sich auseinanderzuentwickeln. Fast unmerklich änderte sich das Wording jetzt von "Drop-in-Replacement" in "compatible". Und heute sind die beiden Datenbanken nicht einmal mehr hundertprozentig kompatibel. Beim Wechsel heißt es jetzt aufpassen!

3.2 Benutzerverwaltung - MySQL bzw. MariaDB

MySQL verwaltet die Benutzer in einer Tabelle (bzw. View ab mariadb 10.4.X) mit dem Namen `user` in der Datenbank mysql. Sie können Benutzer folgendermaßen bearbeiten:

- Anlegen eines neuen Users mit dem `CREATE USER` Statement (Löschen mit `DROP USER`)
- mit `GRANT/REVOKE`-Statements
- durch direkte Veränderung der Berechtigungstabellen (nur bei MySQL); bei MariaDB sind die User in `mysql.global_priv` gespeichert, wobei diese Tabelle nicht direkt bearbeitet werden sollte.

Die bevorzugte Methode ist, zuerst den Benutzer mit CREATE USER anzulegen und anschließend die `GRANT`-Statements zu benutzen, denn sie sind präziser und weniger fehleranfällig[1].

3.2.1 GRANT

Standardmäßig kann außer dem Datenbankadministrator nur derjenige Benutzer ein Datenobjekt bearbeiten, der es auch erstellt hat. Um zusätzliche Zugriffsrechte zu definieren, wird die SQL-Anweisung GRANT verwendet.

[1] Außerdem gibt es eine Menge von Dritten beigesteuerte Programme wie `phpmyadmin`, die benutzt werden können, um Benutzer zu erzeugen und zu verwalten.

Syntax der GRANT-Anweisung:

```
GRANT berechtigung_art [(spalten_liste)] [, ...]
    ON {tabelle | * | *.* | datenbank.*}
    TO benutzername [IDENTIFIED BY 'passwort'][, ...]
    [WITH GRANT OPTION]
```

Bei GRANT und REVOKE-Statements kann berechtigung_art wie folgt angegeben werden:

ALL PRIVILEGES	FILE	RELOAD
ALTER	INDEX	SELECT
CREATE	INSERT	SHUTDOWN
DELETE	PROCESS	UPDATE
DROP	REFERENCES	USAGE

Am Ende der Anweisung kann die Option WITH GRANT OPTION angegeben werden, um dem Benutzer die Möglichkeit der Weitergabe von Rechten zu ermöglichen. [15mm]
Beispiel:

```
GRANT ALL PRIVILEGES ON *.* TO username@localhost
   IDENTIFIED BY 'passwort' WITH GRANT OPTION;
```

Legt einen echten Superuser an, der sich von localhost mit dem Server verbinden kann, aber das Passwort 'passwort' dafür verwenden muss.

3.2.2 REVOKE

Um Benutzern Rechte zu entziehen wird die REVOKE-Anweisung verwendet.
Syntax der REVOKE-Anweisung:

```
REVOKE berechtigung_art [(spalten_liste)] [, ...]
    ON {tabelle | * | *.* | datenbank.*}
    FROM benutzername [, ...]
```

3.2.3 deprecated - MySQL-Berechtigungstabellen

Dieselben Benutzerzugriffsinformationenen können direkt mittels INSERT-Statements eingegeben werden. Anschließend muss der Server die Berechtigungstabellen neu laden.
Beispiel:

```
INSERT INTO user VALUES('localhost','username',PASSWORD('passwort'),
                  'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
FLUSH PRIVILEGES;
```

Anmerkung: Abhängig von der MySQL-Version müssen eventuell eine andere Anzahl von 'Y'-Werten eingeben, da ältere Versionen weniger Berechtigungsspalten haben.
MariaDB: wie bereits oben beschrieben, sind globale Rechte in aktuellen MariaDB Versionen (ab 10.4.X) in einer Tabelle `mysql.global_priv` gespeichert. Hier sind viele Daten als json-String in einer Zelle enthalten, wobei die Rechte als Zahlenwert festgehalten werden. Aus Kompatibilitätsgründen exisitert eine View `mysql.user`, wo diese Daten wieder in das ursprüngliche Format gebracht werden können.

3.3 Aufgabe

Führen Sie die folgenden Übungen auf dem lokalen `XAMPP`-System[2] unter Windows aus:

- Ermitteln Sie, welche Benutzer dem System bekannt sind.
- Legen Sie mittels `phpMyAdmin` einen neuen Benutzer an und kontrollieren Sie die zugehörigen SQL-Statements.

[2] `http://www.apachefriends.org/de/xampp.html`

- Legen Sie mittels GRANT-Statement ebenfalls einen Benutzer an.
- Schreiben Sie eine einfache PHP-Seite, in der Sie mit den eben erstellten Benutzern auf eine Tabelle zugreifen.
- Erstellen Sie 2 Benutzer (UserA, UserB) und 3 verschiedene Datenbanken (DatenbankA, DatenbankB und DatenbankC) und stellen Sie die Berechtigungen so ein, dass
  - UserA nur auf DatenbankA alle Rechte
  - UserB nur auf DatenbankB alle Rechte
  - UserA sowie UserB auf DatenbankC nur Leserechte besitzen
- Fügen Sie als root einige Tabellen inkl. Werte in die 3 Datenbanken ein und kontrollieren Sie danach die Rechte der beiden Benutzer.
- Entfernen Sie alle angelegten Datenbanken und Benutzer vom System.
- Beantworten Sie folgende Fragen:
  - Werden Rechte aufkummuliert bei weiteren GRANT-Befehlen?
  - `with grant option`: Muss GRANT-Recht vergeben werden, um GRANT ausführen zu dürfen?
  - `with grant option`: Darf man das Recht auch weitergeben?
  - Wenn man bei einer Tabelle kein spaltenspezifisches Recht auf den PRIMARY KEY hat, darf man dann Daten einfügen?
  - Wenn man eine VIEW anlegt, dann das insert Recht auf diese View vergibt aber auf der Tabelle kein insert Recht hat, kann man dann Daten einfügen?
  - Wie sollte ein MySQL-Server abgesichert werden, wenn sie dieser in einem Produktivsystem im Einsatz sein soll?
  - Wie kann man sich die vergebenen Rechte für einen Benutzer anzeigen lassen?
  - Rollenkonzept: Testen Sie das erst in den letzten Versionen von MariaDB eingeführte Rollenkonzept (Rollen anlegen, Rechte vergeben, Rechte entziehen, Rollen entziehen.

3.4 Oracle

Für den Unterricht wird ihnen eine vorinstallierte Version eines Oracle Datenbankservers in einer virtuellen Maschine zur Verfügung gestellt. Kopieren Sie sich diese und dann starten Sie die VM.

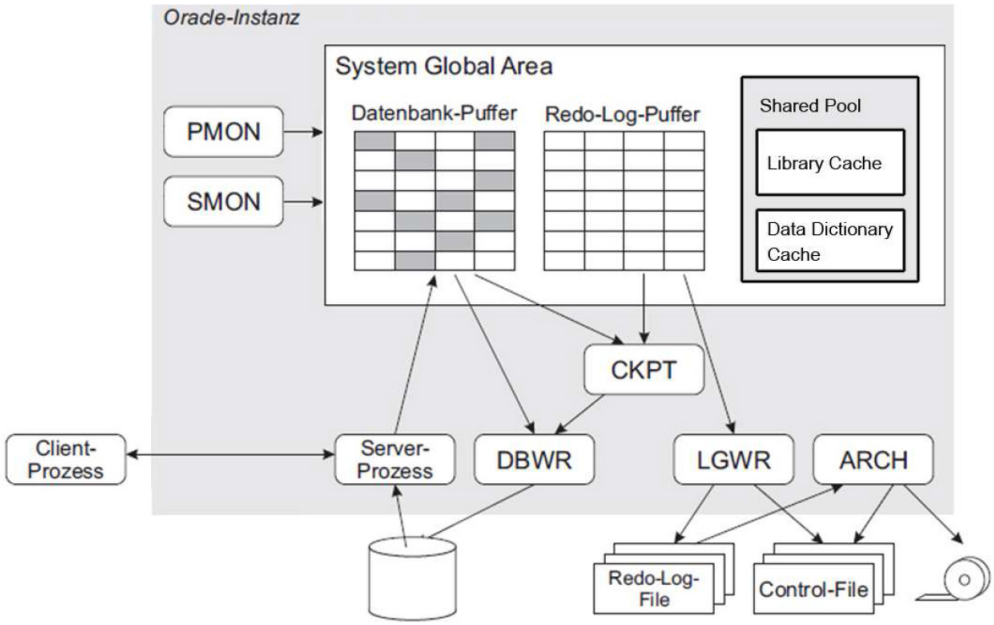

Abbildung 3.1: Architektur von Oracle

3.4.1 Architektur

Ein Oracle-Datenbanksystem, welches von Clients verwendbar ist, besteht aus:

- einem oder mehreren Listener-Prozessen (Oracle-Listener)
- einer oder mehreren Datenbank-Instanzen (Oracle-Instance), das eigentliche Datenbankmanagementsystem (DBMS)
- eine Menge von Datenbank-Dateien (Oracle-Database), die eigentliche Datenbank (DB)

Listener-Prozess (Oracle-Listener)

Nimmt Verbindungswünsche von Datenbank-Clients entgegen und verbindet sie mit einer Datenbank-Instanz. Startet Oracle-Server-Prozesse für Datenbank-Clients.

Datenbank-Instanz (Oracle-Instance)

Hier erfolgt die Ressourcen-Zuteilung für CPU und RAM. Eine Instanz besteht aus mehreren Oracle-Server-Prozessen (Vordergrund- und Hintergrundprozesse), welche den gemeinsamen Arbeitsspeicher in Form von Shared Memory bereitstellen. Die Vordergrundprozesse nehmen Datenbankabfragen (Query) oder Datenmodifikationsanweisungen (DML) in der Sprache SQL von den Datenbank-Clients entgegen, führen diese Aufträge aus und liefern Ergebnisdaten zurück. Dabei agieren die Oracle-Server-Prozesse teilweise direkt auf den Datenbank-Dateien, teilweise übertragen sie jedoch auch Aktivitäten auf die Hintergrundprozesse der Datenbank-Instanz.

Die wichtigsten Hintergrundprozesse sind:

- Der Database Writer (DBWR), der Änderungen an den Datenblöcken in die Data-Files schreibt.
- Der Log Writer (LGWR), der Redo-Informationen in die Redo-Log-Files schreibt.
- Der Archiver (ARCH), der Redo-Log-Files archiviert, sofern die Datenbank im sogenannten Archive-Log-Modus betrieben wird.
- Der System Monitor (SMON), der Konsistenzinformationen in die Control-Files sowie in die Header von Data-Files und in Redo-Log-Files schreibt. Beim Wiederanlauf einer Datenbank nach einem Crash prüft der System-Monitor diese Konsistenzinformationen in einem Quercheck über alle Control-Files, Data-Files und Redo-Log-Files. Sollte der SMON Inkonsistenzen feststellen, so leitet der System-Monitor eine Crash Recovery ein, bei dem aus den Redo-Log-Files solange fehlende Transaktionen in die Data-Files übertragen werden, bis die Datenbank mit allen Data-Files wieder in sich konsistent ist.
- Der Prozess Monitor (PMON), der die Oracle-Prozesse überwacht.

Der Oracle System Identifier (Kurzform: SID) ist die Kennung für die Instanz von Oracle, die auf dem Server läuft. Diese Kennung ist erforderlich, wenn eine Verbindung zu einem Server hergestellt werden soll, der mehr als eine Instanz einer Oracle-Datenbank unterstützt. Die SID wird in network/admin/tnsnames.ora unter dem Oracle-Installationsverzeichnis gespeichert.

Menge von Datenbank-Dateien (Oracle-Database)

Hier werden die Daten gespeichert. Dies erfolgt zumeist in Dateien in einem Dateisystem. Man unterscheidet die Dateiarten:

- Data-Files: die eigentlichen Datendateien mit den Dateninhalten.
- Redo-Log-Files: sehr schnell schreibbare Dateien, die als Transaktionslogs dienen und die die Datenblockänderungen (Change-Vektoren) von Transaktionen aufnehmen. Diese gespeicherten Change-Vektoren dienen zur Wiederherstellung von Datenblöcken, falls ungeplant oder beabsichtigt das Datenbankmanagementsystem terminiert werden muss. Noch nicht in die Data-Files übertragene, festgeschriebene Änderungen können so rekonstruiert und nachgefahren werden (roll forward). Im Anschluss daran werden alle Änderungen nach dem letzten erfolgreichen Festschreibvorgang (check point) zurückgeschrieben (roll back).
- Control-Files: Dateien, die u. a. die Struktur- und Zustandsinformation der Datenbank enthalten. Hierzu zählen die System-Change-Number (SCN) sowie die Pfade und Namen aller Data-Files und Redo-Log-Files.

3.4.2 SQL*PLUS

SQL*PLUS ermöglicht die interaktive Eingabe von SQL-Statements und bietet außerdem noch Möglichkeiten, Abfrageergebnisse zu formatieren, Optionen zu setzen und SQL-Statements zu editieren beziehungsweise zu speichern.

3.4.3 Aufruf und Beenden von SQL*PLUS

SQL*PLUS wird von der Systemumgebung folgendermaßen aufgerufen:

```
sqlplus <username>[/<password>][@database] [as sysdba]
zB:sqlplus user/pass@database
zB:sqlplus sys as sysdba
```

Nach erfolgreichem Login meldet sich SQL*PLUS mit dem Prompt

```
SQL>
```

Nun können folgende Befehle eingegeben werden:

- SQL-Befehle, um auf die Informationen in der Datenbank zuzugreifen. SQL-Befehle werden mit einem ; (Semikolon) abgeschlossen. Der betreffende Befehl wird dann umgehend ausgeführt.
- SQL*PLUS-Befehle, die das Arbeiten mit SQL vereinfachen.

Um SQL*PLUS zu verlassen, verwendet man das EXIT Kommando:

```
SQL> EXIT
```

SQL*Plus-Austausch-Variablen: Wenn man eine Austausch-Variable das erste Mal mit & verwendet, dann wird bei jeder Skriptausführung eine Eingabeaufforderung ausgeführt.

```
SQL> SELECT empno, ename FROM emp WHERE empno = &x;
Geben Sie einen Wert fuer x ein: 7369
alt    1: SELECT empno, ename FROM emp WHERE empno = &x
neu    1: SELECT empno, ename FROM emp WHERE empno = 7369

     EMPNO ENAME
---------- ----------
      7369 SMITH
```

3.4.4 SQL*PLUS-Befehle

Die wichtigsten Befehle sind:

`/:`	bewirkt die Ausführung des Inhalts im Eingabepuffer. In diesem Puffer befindet sich immer das letzte SQL-Kommando. Mittels dieses Befehles kann dasselbe SQL-Kommando mehrmals ausgeführt werden.
`EDIT filename`	ruft einen Editor mit dem File filename auf. Der Inhalt von filename kann nun editiert werden. In Linux wird standardmäßig wird der emacs-Editor verwendet. Wollen Sie lieber mit vi arbeiten, so können Sie die Standardeinstellung ändern, indem Sie in Ihrem Homedirectory ein File namens login.sql anlegen und in dieses File die Zeile `DEFINE_EDITOR=vi` einfügen. Damit wird automatisch beim Start von SQL*PLUS der vi-Editor als Standardeditor gesetzt.
`HELP command`	ruft die Online-Hilfe für den Befehl command auf.
`HOST command.`	Mit diesem Befehl können Shell-Kommandos ausgeführt werden, ohne dass SQL*PLUS verlassen werden muss. z.B. zeigen `host dir` den Inhalt des aktuellen Directory an.
`SET`	Über SET können verschiedene Optionen zur Darstellung der Abfrageergebnisse gesetzt werden. Der Befehl `SET LINESIZE 200` setzt beispielsweise die Ausgabebreite auf 200 und sorgt damit dafür, dass bei Abfragen mit vielen Feldern weniger Zeilenumbrüche durchgeführt werden. Der Befehl `SET PAGESIZE 100` sorgt dafür, dass erst nach 100 angezeigten Zeilen (Tupel) die Spaltenüberschriften (Attributnamen) erneut abgedruckt werden.
`SPOOL filename`	speichert die Ergebnisse von Queries in dem File filename. Dieser Befehl ist solange aktiv, bis das Spooling mit `SPOOL OFF` wieder unterdrückt wird. Solange das Spooling aktiv ist, kann das Ergebnisfile nicht ausgelesen werden.
`START` bzw. `@filename`	führt den Inhalt des Commandfiles filename aus.
`DESCRIBE object`	Liefert Informationen über eine Objekt (Tabelle, View, Funktion, Prozedur, ...). Der Befehl `DESCRIBE table1` liefert beispielsweise sämliche Spaltendefinitionen der Tabelle table1.

3.4.5 Das Command-File

In einem Command-File können mehrere SQL*PLUS- und SQL-Befehle zusammengefasst und dann mit dem SQL*PLUS-Befehl `START` auf einmal ausgeführt werden. Dabei ist folgendes zu berücksichtigen:

1. Das Command-File muss die Extension .sql haben.
2. Die einzelnen Befehle müssen mittels eines ; (Semikolon) am Zeilenende abgeschlossen werden.

   Beispiel:

   ```
   SELECT * FROM table_1; /* Query 1 */
   ...
   SELECT * FROM table_n; /* Query n */
   ```

3. Um die Ergebnisse der Queries im Commandfile in eine Datei umzuleiten, empfiehlt es sich, den SQL*PLUS-Befehl `SPOOL` zu verwenden.

   Beispiel:

   ```
   SPOOL Ergebnisfile      /* Kein ; nach SQL*PLUS-Befehlen!!!*/
   SELECT * FROM table_1; /* Query 1 */
   ...
   ```

```
SELECT * FROM table_n; /* Query n */
SPOOL OFF
```

Nachdem das Command-File mit START ausgeführt worden ist, können die Ergebnisse mittels `EDIT Ergebnisfile` in den Editor geladen werden.
Achtung: Evtl. wurde an das Ergebnisfile automatisch die Erweiterung .lst angehängt.

Ein Command-File kann ebenfalls durch

```
sqlplus -S <username>[/<password>]@database @<commandfile>
```

direkt von der OS-Ebene ausgeführt werden. Dabei wird der SQL*Plus-Editor gestartet und das angegebene Skript ausgeführt. Die Option `-S` unterdrückt die Startmeldungen von SQL*Plus (Copyright, Prompt,...). Weitere Ausgaben lassen sich durch Setzen von Variablen im Skript steuern (z.B. `set feedback off, set verify off`, ...). Als letzter Befehl im Skript muss ein `exit` stehen, da sonst der SQL*Plus-Editor nicht beendet wird. Sinnvoll für weitere Fehlerbehandlung ist der Befehl `whenever sqlerror exit sql.sqlcode` am Anfang des Skripts, der dazu führt, dass SQL*Plus sofort bei Auftreten eines Fehlers verlassen wird.

3.5 Anfragen an das Data Dictionary

Im Data Dictionary sind die sogenannten Metadaten, die Daten über die Struktur der Datenbank und der Objekte enthalten:

- Beschreibung der logischen und der physischen Datenbankstruktur
- Namen der für die Datenbank eingerichteten Benutzerrollen und Privilegien
- Speicherplatzinformationen (Tablespaces)
  Ein Tablespace kann als selbstständige administrative Einheit betrachtet werden. Viele Einstellungen (z.B. Blockgrößen) sowie Operationen (z.B. Datensicherung, Wiederherstellung, Verteilung der Daten auf den Festplatten und die Erteilung von Berechtigungen) basieren auf Tablespaces.

  Für die physische Speicherung der Daten verwendet Oracle Datendateien. Der Zugriff auf die Daten, die logische Verwaltung, erfolgt über Tablespaces. Jeder Tablespace gehört zu einer Datenbank und ein Tablespace kann aus mehreren Datendateien bestehen. Eine Datendatei gehört zu einer Datenbank und genau einer Tablespace.

  Es werden 2 Arten von Tablespaces unterschieden:

  - Tablespace System (für das System)
  - die restlichen Tablespaces (zB users, temp, ...)

  Die Aufteilung in mehrere Tablespace ist sinnvoll, wenn man z.B. Daten in einen anderen Ordner verschieben oder offline sichern möchte. Dazu ist der entsprechende Tablespace offline zu setzen.

```
CREATE [TEMPORARY / UNDO] TABLESPACE <tblspc_name>
 DATAFILE / TEMPFILE
  '<datafile01_name and Path where file to create>' SIZE <integer M>[,
  '<datafile02_name and Path where file to create>' SIZE <integer M>[,
  '<datafile0N_name and Path where file to create>' SIZE <integer M>[,...]]]
 BLOCKSIZE <DB_BLOCK_SIZE parameter /2k/4k/8k/16k/32k >
 AUTOEXTEND { [OFF/ON (NEXT <integer K/M >
  MAXSIZE<integer K/M >) / UNLIMITED] }
 LOGGING/NOLOGGING (Logging default)
 ONLINE/OFFLINE (Online default)
 PERMANENT / TEMPORARY (Permanent default)
 ...

ALTER TABLESPACE tablespacename OFFLINE
```

- Informationen über CONSTRAINTS
- aufgezeichnete Ereignisse (Auditing-Informationen)

Das Data Dictionary besteht aus mehreren Tabellen/Views, die wie üblich durch SELECT ... FROM befragt werden können.

```
SQL> SELECT table_name FROM dictionary;
```

liefert eine Übersicht über die Views des Data Dictionaries.
Die Views unterteilen sich in 3 Kategorien:

- `USER_` liefert Informationen über Objekte des aktuellen Benutzers (Zugriff auf eigene Objekte).
- `ALL_` liefert Informationen über alle Objekte, die der aktuelle Benutzer im Zugriff hat. Dazu gehören Objekte, die der Gruppe PUBLIC zugeordnet wurden sowie Objekte, die explizit gegranted wurden.
- `DBA_` liefert Informationen über alle Objekte mit zusätzlichen Informationen für den DBA.

z.B. ergibt

```
SQL> SELECT object_name, object_type FROM user_objects;
```

die Namen aller gespeicherten Objekte des aktuellen Benutzers aus, und zeigt welchem Datenobjekttyp sie angehören.
Speziell kann man mit

```
SQL> SELECT object_name FROM user_objects WHERE object_type='TABLE' ;
```

oder

```
SQL> SELECT table_name FROM user_tables;
```

die Namen aller Tabellen ausgeben.

3.5.1 Rechtevergabe in Oracle

siehe auch: http://de.wikibooks.org/wiki/Oracle:_Benutzerverwaltung

Die Organisation der Tabellen basiert bei ORACLE auf dem Schema-Konzept. Ähnlich dem Benutzerverzeichnis eines Betriebssystem-Accounts befindet sich der Benutzer nach dem Einloggen in seinem persönlichen Arbeitsbereich (Schema), der gleichzeitig auch der Benutzername `<USERNAME>` ist. Erzeugen von Objekten sowie Anfragen finden unter diesem Schemen-Namen statt. Die Bezeichnung der Tabellen geschieht lokal, d.h. Tabellen werden - über ihren Namen `<TABLE>` angesprochen - im Schema `<USERNAME>` gesucht. Um auf Tabellen eines anderen Benutzers zugreifen, muss vor dem Tabellennamen noch der entsprechende Schemen-Name `<USERNAME>.<TABLE>` angegeben werden. So können mehrere Benutzer Tabellen oder Sichten mit gleichem Namen haben.

Das Gruppen- und Rollenkonzept:

- Zwischen Benutzern der Datenbank und Datenbankobjekten sind die Rechte (Privilegien) angesiedelt, d.h. die Erlaubnis, bestimmte SQL-Anweisungen auf Objekten (Klassen von Objekten) ausführen zu dürfen.
- Der Aufwand bei der Benutzerverwaltung kann durch das Rollenkonzept minimiert und verschiedene Rechte in einer Rolle zusammengefasst werden.
- Rechte innerhalb einer Rolle können einzelnen Benutzern sowie Gruppen von Nutzern mit gleichen Befugnissen zugeteilt werden. So können Rechte nur einmal für eine Gruppe, anstatt für jedes Mitglied einzeln vergeben werden.
- Die Granularität der Rechte reicht nicht bis auf Tupel-Ebene. Rechte können nur an ganzen Tabellen oder Sichten vergeben werden.
- Jeder Benutzer kann in mehreren Gruppen Mitglied sein, wodurch seine Autorisierung die Vereinigung der Rechte aller Einzelgruppen ist, zusätzlich zu den Rechten, die direkt an seine Person gebunden sind.

Einige vordefinierte Rollen stehen bereits nach der Installation von ORACLE zur Verfügung:

- CONNECT:
  - einloggen auf der Datenbank (CREATE / ALTER SESSION),
  - Query's / INSERT / UPDATE / DELETE auf Tables bzw. Views (auch anderer Benutzer - sofern freigegeben),
  - CREATE Table / Cluster / Database Link / Sequence / Synonym / View
- RESOURCE:
  - CREATE Table / Procedure / Sequence / Trigger / Cluster in einem (vom DBA festgelegten) Tablespace / auf gesamter DB;
- DBA:
  Non System DBA:
  - alle CONNECT- und RECOURCE-Rechte,
  - SQL-Access auf alle Datenbank-Objekte, außer denjenigen des Besitzers SYS,
  - Erteilen von GRANT / REVOKE Privilegien an Benutzer,
  - Erstellen von (allen Benutzern zugänglichen) PUBLIC Synonyms,
  - Database Recovery,
  - CREATE, ALTER Tablespace, Rollbacksegmente;

  System DBA:

- CREATE DATABASE,
- STARTUP / SHUTDOWN,
- GRANT im Namen anderer Datenbankbenutzer;

3.5.2 Benutzer/Rollen anlegen

Zum Anlegen von Benutzern und Rollen existiert eine eigene Benutzerverwaltung. Um Rollen erzeugen zu können, muss der Benutzer das Systemprivileg CREATE ROLE sowie Rechte zur Vergabe der Objektprivilegien besitzen. Das Anlegen und Löschen von Benutzern und Rollen erfolgt mit CREATE USER / CREATE ROLE bzw. DROP USER / DROP ROLE:

```
CREATE USER <USER> IDENTIFIED BY <PASSWD>
    DEFAULT TABLESPACE USERS
    TEMPORARY TABLESPACE TEMP;

CREATE ROLE <ROLE>;

GRANT SELECT, INSERT, UPDATE, DELETE ON <TABLE> TO <ROLE>;

GRANT <ROLE> TO <USER>;
```

Ändern von Benutzercharakteristika - wie beispielsweise des Passwortes - erfolgt über ALTER USER:

```
ALTER USER <USER> IDENTIFIED BY <NEWPASSWD>;
```

3.5.3 Systemprivilegien

Systemprivilegien beziehen sich auf Klassen von Objekten und dienen dem Zugriff auf alle Objekte eines Typs. Bestimmte Schemaobjekte wie Cluster, Indices, Trigger werden nur durch Systemberechtigungen gesteuert.
In ORACLE existieren verschiedene Systemberechtigungen, welche i.d.R. vom DBA vergeben werden.:

- CREATE [ANY] TABLE/VIEW/TYPE/INDEX/CLUSTER/TRIGGER/PROCEDURE/DATABASE LINK
  Benutzer darf die [alle] entsprechenden Schema-Objekte erzeugen
- ALTER [ANY] TABLE/TABLESPACE/TYPE/TRIGGER/PROCEDURE
  Benutzer darf die [alle] entsprechenden Schema-Objekte verändern
- DROP [ANY] TABLE/VIEW/TYPE/INDEX/CLUSTER/TRIGGER/PROCEDURE
  Benutzer darf die [alle] entsprechenden Schema-Objekte löschen
- SELECT/INSERT/UPDATE/DELETE [ANY] TABLE
  Benutzer darf [in allen] Tabellen Tupel lesen/erzeugen/verändern/entfernen

ANY - Operation sind in jedem Schema / ohne ANY - nur in eigenem Schema erlaubt; zu beachten ist hierbei, dass bspw., wenn das CREATE ANY TABLE Recht vergeben werden und eine Tabelle mit Keys angelegt werden soll, auch das CREATE ANY INDEX Recht vergeben werden muss.[3].

Rechte vergeben

```
GRANT <privileg-list>
  TO <user-list> | PUBLIC
  [WITH ADMIN OPTION];
```

[3] ANY notierte Operationen sollten nur an Administratoren vergeben werden.

PUBLIC: jeder erhält das Recht (sollte sparsam eingesetzt werden)
ADMIN OPTION: Empfänger darf das Recht weitergeben

Beispiele:

```
GRANT CREATE TABLE, ALTER ANY TABLE, DROP ANY TABLE
  TO <username>;

GRANT CREATE ANY INDEX, DROP ANY INDEX
  TO <username>
  WITH ADMIN OPTION;
```

Rechte entziehen

```
REVOKE <privileg-list> | ALL
  FROM <user-list> | PUBLIC;
```

Beispiel:

```
REVOKE ALTER ANY TABLE, DROP ANY TABLE
  FROM <username>;
```

3.5.4 Objektprivilegien

Objektprivilegien berechtigen dazu, Operationen auf existierende Objekte auszuführen:

- Datenmanipulation - DML (Data-Manipulation-Language)
  - INSERT INTO object - Erzeugen einer / mehrerer neuer Zeilen in einer Tabelle
  - DELETE FROM object - Löschen einer / mehrerer Zeilen aus einer Tabelle
  - UPDATE object - Verändern einer / mehrerer Zeilen einer Tabelle

  Einschränken des INSERT- bzw. UPDATE-Privilegs für Spalten ist möglich. Die Spalten ohne INSERT-Privileg werden dann mit null-/ default-Werten belegt. Bei DELETE erfolgt natürlich eine Fehlermeldung.
- Datendefinition - DDL (Data-Definition-Language)
  - CREATE object Erzeugen neuer Objekte
  - ALTER object Verändern der Eigenschaften existierender Objekte
  - DROP object Löschen existierender Objekte
- REFERENCES ermöglicht es anderen Usern, Abhängigkeiten zu Tabellen festzulegen.
- Das EXECUTE Privileg kann für spezielle Prozeduren, Funktionen oder Packages vergeben werden, um Prozeduren, Funktionen bzw. Trigger auszuführen.
  Anwender einer Prozedur benötigen nur das EXECUTE-Privileg, Eigentümer die Systemprivilegien zum Erzeugen der Prozedur (CREATE PROCEDURE / CREATE ANY PROCEDURE / ALTER ANY PROCEDURE) sowie die Objektprivilegien für UPDATE / DELETE auf die betreffenden Tabellen zum Ausführungszeitpunkt.

Rechte vergeben

```
GRANT <privileg-list> | ALL
      [<column-list>]
      ON <object>
      TO <user-list> | PUBLIC
       [WITH GRANT OPTION];
```

ALL: alle Privilegien, die man an dem Objekt hat, werden weitergegeben GRANT OPTION: Empfänger darf das Recht weitergeben PUBLIC: jeder erhält das Recht `<object>` kann sein von der Art TABLE / VIEW / PROCEDURE / FUNCTION / PACKAGE / TYPE `<privileg-list>` ist eine Aufzählung der Privilegien, die man erteilen will, wie

- CREATE / ALTER / DROP für die verschiedenen Arten von Schemaobjekten (DDL Operationen),
- INSERT / DELETE / UPDATE / SELECT für Tabellen und Views (DML Operationen),
- INDEX / ALTER / REFERENCES für Tabellen.

`<column-list>` Einschränkung von Zugriffen auf Spalten von Tables/Views für INSERT / UPDATE / REFERENCES

- Einschränkung auf Spalten möglich,
- mit UPDATE weder löschen noch einfügen möglich,
- bei DELETE erfolgt eine Fehlermeldung.

Beispiel:

```
GRANT INSERT(<col1>,<col3>,<col8>),UPDATE(<col3>,<col8>),SELECT
      ON <TABLE>
      TO <USER>;
```

Rechte entziehen

```
REVOKE <privileg-list> | ALL
       ON <object>
       FROM <user-list> | PUBLIC
        [CASCADE CONSTRAINTS];
```

3.6 Aufgaben

1. Installieren Sie Oracle in einer virtuellen Maschine.
2. Erstellen Sie ein passendes Benutzerprofil für die zukünftigen Benutzer:
   - Ressourcenbeschränkungen einstellen (wie oft kann sich ein Benutzer gleichzeitig anmelden (Sessions), wie viel CPU-Kapazität kann er in einer gewissen Zeit verbrauchen)
   - Passwortmanagement durchführen (Passwortlifetime, Passwortgracetime - wann muss Benutzer Passwort ändern, Passwortlocktime - wie lange ist Benutzer gesperrt, wenn er sich falsch anmeldet)

   Profil erstellen:

   ```
   CREATE PROFILE <Profilname>
       LIMIT [<Ressourcenbeschraenkungen>] [<Passwortmanagement];
   ```

   Beispiel:

   ```
   CREATE PROFILE schueler LIMIT FAILED_LOGIN_ATTEMPTS 3
   PASSWORD_LOCK_TIME 1
   PASSWORD_LIFE_TIME 30
   PASSWORD_GRACE_TIME 5;
   ```

   Profil löschen:

   ```
   DROP PROFILE <Profilname>;
   ```

   Abfragen, welche Profile derzeit existieren (vordefinierte View DBA_PROFILES):

   ```
   SELECT * FROM DBA_PROFILES;
   SELECT DISTINCT PROFILE FROM DBA_PROFILES;
   ```

3. Erstellen Sie einen Beispielbenutzer:

   Benutzer erstellen:

   ```
   CREATE USER <Benutzername> IDENTIFIED BY <Password>|EXTERNALLY
   [DEFAULT TABLESPACE <TableSpaceName>]
   [TEMPORARY TABLESPACE <TempTableSpaceName>]
   [QUOTA <Groesse>[K|M]|UNLIMITED ON <TableSpaceName> ...]
   [PROFILE <ProfilName>]
   [PASSWORD EXPIRE]
   [ACCOUNT LOCK|UNLOCK];
   ```

   PASSWORD EXPIRE: Das Passwort muss bei der ersten Anmeldung geändert werden.

   Beispiel:

   ```
   CREATE USER gerber IDENTIFIED BY "htl"
   DEFAULT TABLESPACE USERS
   PROFILE schueler
   PASSWORD EXPIRE;
   ```

   Abfragen, welche Benutzer derzeit existieren (vordefinierte View DBA_PROFILES):

   ```
   select username from dba_users;
   ```

4. Rechte für den angelegten Benutzer vergeben und die korrekte Funktion testen:

   ```
   GRANT CONNECT TO gerber;
   ```

   Melden Sie sich als gerber an und versuchen Sie eine Tabelle anzulegen

```
sqlplus gerber@demodb

CREATE TABLE Tab1 (
  a int PRIMARY KEY
);
```

Keine ausreichenden Berechtigungen? Seit Oracle 10g Release 2 besitzt die CONNECT-Rolle nur mehr das CREATE SESSION-Recht, alle anderen Privilegien wurden entfernt. Daher sollte besser eine eigene Rolle definiert werden.

5. Erzeugen Sie ein Commandfile `cr_ro_schueler.sql` mit folgendem Inhalt.

```
--Rolle erzeugen
create role ro_schueler;

grant create table     to ro_schueler;
grant create view      to ro_schueler;
grant create procedure to ro_schueler;
grant create trigger   to ro_schueler;
grant create sequence  to ro_schueler;

-- statt Connect-Rolle
grant create session   to ro_schueler;
```

Führen Sie dieses Commandfile aus.

6. Erzeugen Sie ein Commandfile `cr_schueler.sql` mit folgendem Inhalt.

```
create user &1 identified by &2;
grant ro_schueler to &1;
alter user &1 quota 10M on users;
```

7. Erzeugen Sie ein Commandfile `cr_users.sql` mit folgendem Inhalt.

```
-- schueler anlegen
start cr_schueler franz    htl
start cr_schueler otto     htl
start cr_schueler verena   htl
start cr_schueler maria    htl
start cr_schueler peter    htl
start cr_schueler paul     htl
start cr_schueler isabella htl
```

Führen Sie dieses Commandfile aus.

8. Nun sollen alle Benutzer wieder gelöscht werden. Die Benutzernamen sollen nicht manuell eingegeben werden, sondern automatisch ausgesucht werden: Alle Benutzer, welche die Rolle `ro_schueler` zugewiesen haben.

   Erstellen Sie dazu ein Commandfile `dr_users.sql` mit folgendem Inhalt.

```
set pagesize 0
set echo off
set feedback off

spool dr_schueler.sql

select 'drop user '||grantee||' cascade;'
from   dba_role_privs
where  granted_role='RO_SCHUELER';

spool off
```

```
start dr_schueler.sql
```

Führen Sie dieses Commandfile aus.