Kurzeinführung in Oracle

Wolfgang May
may@informatik.uni-goettingen.de
Institut für Informatik
Universität Göttingen

15. April 2026

1 Allgemeines

1.1 Administratives

Das Praktikum wird in der Regel in Gruppen zu 4 Personen durchgeführt. Für das Arbeiten mit ORACLE ist außer einem **Account** im CIP-Pool der Informatik eine Zugangsberechtigung zur Datenbank, ein **Oracle-Account**, erforderlich, mit dem eine ORACLE-Session eröffnet, d.h. die Verbindung zu einer Datenbank hergestellt werden kann. Im Praktikum hat jeder Teilnehmer einen eigenen ORACLE-Account.

Gruppenverzeichnis

Für jede Praktikumsgruppe existiert ein Verzeichnis

`/afs/informatik.uni-goettingen.de/course/db-prakt/grp`n,

für das alle Gruppenmitglieder (sowie die Betreuer) Lese- und Schreibrecht besitzen. Dort sollen die SQL-Skripte der Gruppe abgelegt werden.
Zur Kommunikation stehen die Mail-Aliase

- `dbp-all@informatik.uni-goettingen.de` : Betreuer und Teilnehmer
- `dbp-betr@informatik.uni-goettingen.de` : Betreuer

zur Verfügung. Folienkopien, Übungsblätter etc. werden auf

`http://www.dbis.informatik.uni-goettingen.de/Teaching/DBP/`

bereitgestellt.

1.2 Rechner

CIP-Pool: Als Basis dient der CIP-Pool im Institut für Informatik. Dort sollte jeder Studierende bereits einen Account aus Info I/II etc haben. Die gesamte im weiteren beschriebene Software ist dort installiert.

Wenn die Aufgaben vor Ort im CIP-Pool testiert werden, ist es grundsätzlich sinnvoll, den dortigen Oracle-Account einzurichten.

Erreichbarkeit des CIP-Pools von aussen: mit

`shell.stud.informatik.uni-goettingen.de`

und eine eduVPN-Verbindung (siehe

`https://wiki.student.uni-goettingen.de/support/wlan/vpn_eduvpn`

) und 2.Faktor (das alte

`ssh login.stud.informatik.uni-goettingen.de`

aber auch nur mit eduVPN könnte auch noch gehen) und dann bitte weiterloggen auf einen der "echten" Poolrechner cipXXX.

Eigene Rechner: Oracle kann auch auf einem eigenen Rechner installiert werden. Andere SQL-Datenbanksysteme (postgres, mySQL, ...) benutzen teilweise (ab Versuch 4: PL/SQL, objektrelationale Erweiterungen) eine andere Syntax.

1.3 Modifikationen an der Linux-Umgebung im CIP-Pool

.bashrc/ global anpassen

Die globalen Pfade für Oracle werden über ein Skript initialisiert. Damit dieses atomatisch beim Einloggen gestartet wird, muss man das folgende in die eigene Datei `.bashrc` reinschreiben:

```
source /afs/informatik.uni-goettingen.de/group/dbis/public/oracle/.oracle_env
```

Im einzelnen enthält dieses Skript (Stand April 2022; getestet April 2026) die folgenden Initialisierungen, die man ggf. bei einer eigenen Installation anpassen muss (indem man die entsprechende(n) Zeile(n) auch in das eigene .bashrc einträgt und ändert):

```
# Oracle path ins AFS
export ORACLE=/afs/informatik.uni-goettingen.de/group/dbis/public/oracle
export ORACLE_HOME=$ORACLE/instantclient
export TNS_ADMIN=$ORACLE/etc
export TWO_TASK=dbis
export PATH=$ORACLE_HOME:$ORACLE/bin:$ORACLE/sqlj/bin:$PATH
export LD_LIBRARY_PATH=$ORACLE_HOME:$ORACLE/lib:$LD_LIBRARY_PATH
export CLASSPATH=.:$ORACLE_HOME/ojdbc8.jar:$ORACLE/sqlj/lib/runtime12.jar:
                 $ORACLE/sqlj/lib/translator.jar:$CLASSPATH
# sqlplus alias with history
test `which rlwrap` && alias sqlplus > /dev/null 2>&1 ||
                 alias sqlplus="rlwrap sqlplus"
```

Konfiguration der eigenen Umgebung:

- Benutzerspezifische Konfiguration der Oracle-Sessions:
  - Die Bildschirmausgabe von SQL*Plus wird über (Session-)Parameter konfiguriert. Diese werden in der Datei `oracle/login.sql` gesetzt, die automatisch aufgerufen wird, wenn SQL*plus gestartet wird. Um z.B. nach jeweils 50 Bildschirmzeilen auf eine Benutzereingabe zu warten, sollte diese Datei die folgenden Zeilen enthalten:

`login.sql`

```
set pause '- continue -'
set pause on
set pagesize 50
```

Entsprechend kann man mit `set linesize 200` die Zeilenlänge beliebig einstellen. (Die Bedeutung dieser (und weiterer) Variablen kann man sich in SQL*Plus (siehe unten) mit `help set` anschauen).

- Die Sprache, in der Oracle-Messages ausgegeben werden, wird über die Umgebungsvariable `NLS_LANG` gesteuert. Sprache

  `export NLS_LANG=AMERICAN_AMERICA.UTF8` für (amerikanisches) Englisch und
  `export NLS_LANG=GERMAN_GERMANY.UTF8` für Deutsch.

  Was gerade als NLS_LANG (etc.) eingestellt ist, kann man ausgeben lassen:

```
SQL> HOST echo $NLS_LANG
AMERICAN_AMERICA.UTF8
```

- Benutzerkonfiguration auf Linux-Ebene:

  - Die Variablen `ORACLE_PATH` (für SQL*Plus, s.u.) bzw. `SQLPATH` (für SQLcl, s.u.) geben an, wo nach der Anlaufdatei "`login.sql`" sowie nach SQL-Skripten gesucht wird. Es ist sinnvoll, beides in `.bashrc` z.B. mit

```
export ORACLE_PATH=.:/afs/informatik.uni-goettingen.de/
           group/dbis/public/Mondial:$HOME/oracle
export SQLPATH=.:/afs/informatik.uni-goettingen.de/
           group/dbis/public/Mondial:$HOME/oracle
```

    zu setzen (jeweils in einer Zeile); in

```
/afs/informatik.uni-goettingen.de/group/dbis/public/Mondial
```

    befinden sich die Skripten zum Erstellen der Datenbasis.

  - Weiterhin kann man noch ein Verzeichnis `oracle/` im eigenen Home-Verzeichnis erzeugen, in dem SQL-Skripte und Dateien abgelegt werden und es in den `ORACLE_PATH` aufnehmen. ~/oracle/ erzeugen

  - Wenn man Veränderungen an .bashrc vorgenommen hat, muss man es mit "`source .bashrc`" neu ausführen lassen umd die Änderungen wirksam werden zu lassen.

Weitere Umgebungsvariablen Eine Beschreibung aller möglichen Umgebungsvariablen findet man in der Oracle-Dokumentation ("Administrator's Reference for UNIX-Based Operating Systems"). Links zur Oracle-Dokumentation finden sich auf der Praktikumsseite.

2 Tool: Oracle SQL Developer Command Line (SQLcl)

Oracle SQL Developer Command Line (SQLcl) (siehe Webseite) ist im CIP-Pool installiert und kann auch auf dem eigenen Rechner installiert werden. Es bietet (seit 2021?) ein shell-basiertes interaktives Benutzerinterface, das einen größeren Funktionsumfang bietet, als (das

schon lange existierende – seit (mindestens) 1992) SQL*Plus (siehe unten). Man ruft ihn einfach mit "sql" von der Kommandozeile auf, er fragt dann nach username und passwort. Am besten setzt man (wieder im .bashrc) ein alias:

```
alias sqlcl='sql username/passwort'
```

3 Das SQL-Interface SQL*Plus ("InstantClient")

SQL*Plus (siehe Webseite) ist im CIP-Pool installiert und kann auch auf dem eigenen Rechner installiert werden. Es bietet ein shell-basiertes interaktives Benutzerinterface (auch als "Instant Client" bezeichnet).

SQL starten

SQL*Plus wird mit `sqlplus` in einer Shell aufgerufen. Man wird dann zur Eingabe des Benutzernamens aufgefordert. Danach gibt man das Passwort ein, welches initial dem Benutzernamen entspricht. Nach einigen anderen Anzeigen erscheint als Prompt `SQL>`. Mit `quit` wird SQL*Plus wieder verlassen.

Wichtiger Hinweis: Das Passwort sollte nach dem ersten Einloggen sofort mit dem Befehl

```
passw
```

geändert werden (fragt erst noch einmal nach dem alten, und dann zweimal nach dem neuen Passwort)! Das Passwort muss mit einem Buchstaben beginnen und darf neben Buchstaben und Zahlen nur die Zeichen $, _ und # enthalten.
Es ist sinnvoll, in der Datei `.bashrc` einen alias zu definieren:

```
alias sqlplus='sqlplus <login>/<password>'
```

Noch komfortabler ist

```
alias sqlplus='rlwrap sqlplus <login>/<password>'
```

da man dann gleich auch eine History hat (d.h., mit "Cursor up" bekommt man den letzten Befehl bzw. die zuletzt eingegebene Zeile wieder).
Nach einem "`source .bashrc`" genügt der einfache Aufruf `sqlplus`, es muss nun nichts mehr eingegeben werden.

3.1 Interaktiv

```
SQL>  select * (Return)
  2   from <tabelle>; (Return)
```

Anstatt die Kommandos direkt einzugeben, kann man sie auch per Maus aus einem Editor kopieren.

SQL*Plus im emacs. Wenn man SQL*Plus wie oben beschrieben im xterm aufruft, ist die Oberfläche sehr unkomfortabel (keine History etc). Eine bessere Bedienung ist möglich, wenn man SQL*Plus im emacs laufen lässt: Dort startet man mit `Meta-x shell` (`ALT+x shell` oder `ESC und dann x shell`) eine shell, und ruft darin `sqlplus` auf. Der Vorteil daran ist, dass man so schöne Dinge wie Meta-p/Meta-n (history), Ctrl-a/Ctrl-e (Zeilenanfang/-ende) und Cursortasten benutzen kann. Allerdings wird hier bei der Eingabe das Passwort in Klartext angezeigt, man sollte also wie oben beschrieben einen Alias definieren.

Änderungen. In jedem Fall ist zu beachten ist, dass Änderungen durch SQL-Statements in der Datenbank zwar sofort „sichtbar“ sind, aber nur für denjenigen, der den Befehl ausgeführt hat. Für andere Benutzer sind die Änderungen erst nach Eingabe von `commit` vorhanden (siehe auch Transaktionen). Darüber hinaus werden andere Benutzer gesperrt, wenn sie schreibend auf denselben Tabelleninhalt zugreifen wollen.

3.2 SQL*Plus über SQL-Skripte

SQL-Skripte sind Dateien, die SQL- und SQL*Plus-Statements enthalten, so wie sie auch in SQL*Plus interaktiv eingegeben werden. Die Dateien sollten die Endung *.sql haben (und benötigen kein x-flag, da sie nicht ausgeführt, sondern nur gelesen werden).
Mit ihnen können mehrere Befehle an die Datenbank geschickt werden, ohne diese einzeln in SQL*Plus eingeben zu müssen. Dadurch sind komplexe Transaktionen möglich und man kann Änderungen leicht im Skript durchführen, ohne alle Statements einzeln zu wiederholen.

Editieren per SQL*Plus und eingestelltem Editor.
SQL-Skript editieren
Editor einstellen
Wenn man aus SQL*Plus direkt in einen Editor wechseln will, um SQL-Skripte zu schreiben, kann ein Editor in der Umgebungsvariable `EDITOR` definiert werden (etwa emacs oder xemacs (sollte man können, braucht man immer wieder), nano (klein und handlich), oder vi (Geschmackssache)). Dies kann durch

```
export EDITOR='emacs -nw'
```

in `.bashrc` erreicht werden. In SQL*Plus wird eine Datei durch `edit` <*filename*> in den Editor geladen.
Von SQL*Plus aus wird der eingestellte Editor mit dem Befehl `edit <filename.sql>` zum Bearbeiten der entsprechenden Datei aufgerufen.
Es öffnet sich ein neues Fenster, in dem das gewünschte File editiert werden kann. Nach Verlassen des Editors befindet man sich wieder in SQL*plus.
Skript ausführen
Die Ausführung der Befehle erfolgt durch Eingabe von `start <filename>` (kürzer geht auch `@<filename>`).

```
SQL>  edit name.sql   Aufruf des Editors
  ⋮
SQL>  start name      Ausführen der Datei
```

Man kann natürlich stattdessen auch den (x)emacs mit einem eigenen Fenster öffnen, SQL*Plus vom xterm aufrufen, und Kommandos mit der Maus zwischen den Fenstern kopieren; damit spart man sich das Wechseln.

Nützliche SQL*plus-Kommandos:

ed(it) <file>: startet den eingestellten Editor mit `<file>`

sta(rt) <file>: führt das angegebene SQL-Skript aus. Die Endung `.sql` muss nur angegeben werden, falls der Filename selber Punkte enthält (aus diesem Grund sollte man Skripte mit `lsg1_2.sql` bezeichnen anstatt mit `lsg1.2.sql`).

@<file>: äquivalent zu `start <file>`.

desc(ribe) <table>: zeigt die Tabellenstruktur von `<table>`.

show <xy>: zeigt den momentanen Inhalt der SQL*Plus-Variable `<xy>` an.

show all: zeigt den Inhalt aller SQL*Plus-Variablen an.

help: aktiviert die Hilfefunktion.

help <Stichwort>: aktiviert die Hilfefunktion zu dem angegebenem Stichwort.

ho(st) <cmd>: führt das Linux-Kommando `<cmd>` aus.

! <cmd>: analog zu `host`.

exit oder quit: Beenden des Editors. Dabei wird ein `commit` ausgeführt. Die sonst übliche Unterscheidung zwischen `exit` und `quit` gilt hier nicht. Sollen die letzten Änderungen verworfen werden, ist ein explizites `rollback` notwendig. Wurden SQL*Plus-Variablen verändert, so bleiben diese Änderungen erhalten. Ist dies nicht erwünscht, dann müssen die entsprechenden Variable im File `login.sql` definiert werden, das bei jedem Aufruf von SQL*Plus ausgeführt wird.

Ein SQL-Skript kann ebenfalls durch

```
sqlplus -S @<SQL-Script>
```

direkt von der Linux-Ebene ausgeführt werden. Dabei wird der SQL*Plus-Editor gestartet und das angegebene Skript ausgeführt. Man muss allerdings einen Alias `sqlplus` definiert haben (s.o.). 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.

4 InstantClient SQL*Plus auf dem eigenen Rechner

- Man kann sich den SQL*Plus auch auf dem eigenen Rechner installieren: Von `https://www.oracle.com/database/technologies/instant-client/downloads.html` die Pakete “Basic Package (ZIP)” und “SQL*Plus Package (ZIP)” downloaden und entpacken und Pfadvariablen setzen:
  - Linux: auspacken, dann `LD_LIBRARY_PATH` setzen
  - Falls dann noch das vermisst wird: `apt-get install libaio1`
  - Alias für sqlplus mit Passwort setzen.
  - Normalerweise würde der Instant Client zu einer lokalen Oracle-Instanz verbinden. Um zum IFI-Oracle zu verbinden muss man ihm die notwendigen connection-Daten mitteilen. Dies geschieht entweder wie am IFI über `TNS_ADMIN =/afs/informatik.uni-goettingen.de/group/dbis/public/oracle/etc/tnsname` wo SQL*Net-Konfigurationsdaten abgelegt sind, oder indem man es in das Alias mit reindefiniert:

```
alias sqlplus='rlwrap sqlplus may/passwd@"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
    (HOST=oracle19c.informatik.uni-goettingen.de)(Port=1521))
    (CONNECT_DATA=(SERVICE_NAME=dbis.informatik.uni-goettingen.de)))"'
```

(alles in einer Zeile, Stand Juni 2022).

– Linux: `export NLS_LANG` setzen wie oben beschrieben.

5 Verwendung von Entwicklungsumgebungen

Aus Entwicklungsumgebungen lassen sich ebenfalls SQL- (und teilweise auch PL/SQL)-Statements verschicken. Meistens wird dabei JDBC verwendet, um eine Verbindung zur Datenbank herzustellen.
Hinweis: Die Skripte `create.sql` und `consult.sql` können aus lokalen Entwicklungsumgebungen nicht aufgerufen werden, da sie im IFI-Filesystem liegen.

5.1 Oracle SQL Developer

Die grafische Entwicklungsumgebung "SQL Developer" ist eine Alternative zu SQL*Plus mit vielen Funktionalitäten:

- automatische Code-Formatierung (Auto-Indent),
- automatische Code-Vervollständigung (Code-Completion),
- einen Schema-Browser (zeigt alle Tabellen, Views, etc. die man besitzt),
- man kann sich den intern optimierten Algebra-Baum ("Explain Plan") anzeigen lassen,
- PL/SQL-Debugger.

Im CIP-Pool der Informatik ist der SQL Developer installiert und kann mit

```
/afs/informatik.uni-goettingen.de/group/dbis/public/sqldeveloper/sqldeveloper.sh
```

aufgerufen werden.

- Man kann sich den SQL Developer auch auf dem eigenen Rechner installieren: Von `https://www.oracle.com/downloads/` das Paket "SQL Developer" downloaden und entpacken.
  Man kann sich damit am IFI-Oracle einloggen, ohne im VPN sein zu müssen (April 2023).

Bei regelmäßiger Verwendung kann man sich hierzu einen passenden Alias setzen.

Man muss eine Datenbank-Verbindung erstellen, welche man für den späteren Gebrauch speichern kann. Erforderliche Parameter:

- Hostname: oracle19c.informatik.uni-goettingen.de
- Port: 1521

- SID: *nichts, wird seit Version 12 nicht mehr verwendet, stattdessen ...*
- Service-Name: dbis.informatik.uni-goettingen.de

(ist im Pool (evtl.) auch verfügbar, wenn man unter "Verbindungstyp" TNS aus dem Pulldown-Menu auswählt, und unter "Network Alias" "DBIS" auswählt). Dann

- Connection Name: ein frei wählbarer Name für die Verbindung
- Username: Benutzername
- Password: Passwort

eintragen, und "Abbrechen" (!) klicken.
Alle anderen Parameter können die Default-Werte behalten.
Man kann dann SQL und auch PL/SQL direkt ausführen.

Man kann damit SQL-Skripten aus den lokalen Verzeichnissen ausführen:

- Wenn man auf einem CIP-Pool-Rechner ist:

  ```
  @/afs/informatik.uni-goettingen.de/group/dbis/public/Mondial/create.sql
  ```

  Das geht von zuhause natürlich nicht, weil man dort keinen Zugang auf die Verzeichnisse am IFI hat. Dort geht (z.B. unter Windows) wenn man es passend kopiert hat

  ```
  @D:\Scripts\create.sql
  ```

5.2 Mit IntelliJ

(analog auch für eclipse)

5.2.1 IntelliJ Community Version mit (Linux: buggy) Plugin

SQL-Plugin installieren, z.B. `https://plugins.jetbrains.com/plugin/1800-database-navigator`. Laut `https://confluence.jetbrains.com/display/CONTEST/DBN+-+Change+Notes` wird dieses plugin seit 2015 nicht mehr weiterentwickelt. Neues File `test.sql` anlegen. Daran erkennt IntelliJ, dass man eine DB-Connection verwenden möchte. Oben erscheint "DBConnections", dann erzeugt man eine solche:

- Name, Description: ein frei wählbarer Name für die Verbindung
- Host: oracle19c.informatik.uni-goettingen.de
- Port: 1521
- Database: dbis.informatik.uni-goettingen.de
  und auf "Service name" einstellen, SID wäre falsch.

- User und Passwort: eintragen.

(ist im Pool (evtl.) auch verfügbar, wenn man unter "Verbindungstyp" TNS aus dem Pulldown-Menu auswählt, und unter "Network Alias" "DBIS" auswählt). Dann

- Username: Benutzername
- Password: Passwort

Damit kann man Anfragen und (einige) DDL-Kommandos ausführen. Wenn man damit PL/SQL oder CREATE TABLE ausführen will (das geht nicht direkt über die JDBC-Verbindung), verlangt IntelliJ, SQLplus zu installieren (siehe oben, aber dort muss man nichts konfigurieren), und verwendet dann SQLplus über einen Kommandozeilenaufruf.

- (Stand Juni 2022: funktioniert unter Windows; aber nicht unter Linux, Das Plugin scheint den Aufruf nicht korrekt zu übergeben.

5.2.2 IntelliJ Ultimate Version

Mit der IntelliJ Ultimate-Version (Studierende können sich online eine kostenlose Lizenz besorgen) funktioniert es ohne dass man irgendwas zusätzliches installieren muss. (Noch nicht selber getestet).

6 How to get started – the first session

6.1 Datenbank erzeugen

> **Hinweis:** Dies funktioniert nur mit dem Kommandozeilen-Tool SQL*Plus, da dazu Skripte aus dem Filesystem aufgerufen werden!

Zuerst loggt man sich auf einem Rechner des Informatik-CIP-Pools ein. Dann meldet man sich mit

```
sqlplus
```

über das SQL-Interface auf dem Oracle-Account ein. Man wird dann zur Eingabe des Benutzernamens (entspricht dem Nachnamen) und des Passwortes aufgefordert, welches initial dem Benutzernamen entspricht. Nach einigen anderen Anzeigen erscheint als Prompt `SQL>`. Dann ändert man mit dem Befehl

```
passw
```

das Passwort. Das Passwort muss mit einem Buchstaben beginnen und darf neben Buchstaben und Zahlen nur die Zeichen $, _ und # enthalten. Zu Beginn ist der Oracle-Account noch leer, es muss erst die Datenbasis erzeugt werden. Vorrausgesetzt, die Umgebungsvariable `ORACLE_PATH` enthält `/afs/informatik.uni-goettingen.de/group/dbis/public/Mondial`, geschieht dies in SQL*Plus wie folgt.

Datenbank erzeugen

```
@create.sql;
```

Dadurch wird folgendes SQL-Skript ausgeführt:

```
/afs/informatik.uni-goettingen.de/group/dbis/public/Mondial/create.sql  .
```

Dieses Skript ruft mehrere Unterskripten auf und erstellt alle Tabellen. Bei jedem weiteren Einloggen findet man die Datenbasis so vor, wie man sie zuletzt verlassen hat.
Für den Fall, dass man versehentlich etwas an der Datenbasis löscht/verändert, kann man jederzeit mit `@create.sql` die ursprüngliche Datenbasis wieder erstellen.
Einzelne Tabellen können auch einfacher durch das Skript `@consult` aus der Referenzdatenbasis des Benutzers "dbis" kopiert werden. Um die Schlüsseleigenschaften dabei nicht zu verletzen müssen erst alle Tupel aus der eigenen Tabelle entfernt werden:

```
delete from <tabelle>;
@consult;
Tabelle: <tabelle>
```

Ach ja, jeder SQL-Befehl muss mit einem **Semikolon** abgeschlossen werden ... sonst geschieht nichts!

6.2 Anfragen an das Data Dictionary.

Im Data Dictionary sind Daten über das Datenbankschema (*Metadaten*) gespeichert. Das Data Dictionary besteht aus mehreren Tabellen, die wie üblich durch `SELECT ... FROM` befragt werden können. Mit

```
SELECT * FROM user_objects;
```

erhält man einen Überblick, was so alles gespeichert ist. Z.B. ergibt

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

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

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

die Namen aller Tabellen ausgeben.
Mit

```
DESCRIBE <tabelle>;
```

kann man sich die komplette Definition der Tabelle <`table`> geben lassen; die Tabelle selber kann man sich dann mit

```
SELECT * FROM <tabelle>;
```

anschauen.
Aufgabe: Schauen Sie sich mal in MONDIAL um.

6.3 Transaktionen und Persistente Veränderungen.

Interaktiv vorgenommene Veränderungen am Datenbankinhalt können entweder a) mit `commit` persistent gemacht werden, oder mit b) `rollback` rückgängig gemacht werden. Verlassen von SQL*Plus führt ein automatisches `commit` aus. Der folgende Ablauf veranschaulicht dies:

`select * from city` gibt die gesamte Relation aus.
`delete from city` löscht die Relation, wie man sich mit
`select * from city` überzeugen kann.
`rollback` macht die Veränderung rückgängig, wie man mit einem neuen
`select * from city` sieht.
`select * from country` gibt diese Relation aus.
`delete from country` löscht die Relation, was dann durch
`commit` engültige Wirkung erhält:
`select * from country` . Da hilft auch kein
`rollback` mehr:
`select * from country` . Mit
`delete from sea` und versehentlichem
`quit` gehen auch diese Daten noch ex, wie man nach
`sqlplus` und
`select * from sea` feststellen kann. Jetzt hilft nur noch
`@create.sql` , um die ursprüngliche Datenbank wieder zu bekommen.

Hinweis: DDL-Kommandos machen ein automatisches `commit`.

6.4 Die tägliche Arbeit

Um die Aufgaben zu lösen, ist es sinnvoll, mit `cd $WORK` in das Gruppenverzeichnis zu wechseln, und dort SQL*Plus, (x)emacs, nano usw. aufzurufen. Innerhalb der Gruppe sollte man sich über die Namensgebung der SQL-Skripts einig sein.
Die SQL-Anweisungen sollten in den Skripten einigermaßen strukturiert werden, um die Lesbarkeit zu erleichtern. Schlüsselwörter sollten groß, Tabellennamen klein geschrieben werden:

```
SELECT * FROM city
WHERE country='D'
```

6.5 Hilfsskripte zum Löschen von Schemaobjekten

Es gibt Hilfsskripte, mit denen man alle Objekte einer bestimmten Art löschen kann. Diese Skripte heißen `drop-all-<obj>` (und liegen unter `/afs/informatik.uni-goettingen.de/group/dbis/public/Mondial`, wo auch das o.g. `create`-Skript liegt). Der Aufruf

- `@drop-all-tables`

löscht alle Tabellen, die man besitzt. Außerdem gibt es

- `drop-all-procedures`,
- `drop-all-functions`,
- `drop-all-triggers`,

- `drop-all-types`,
- `drop-all-views`.

6.6 Hilfsprozedur zum Beenden aller Sessions

Es kann vorkommen, dass man Datenbank-Sessions geöffnet hat, auf die man aber nicht mehr zugreifen kann. Das kann z.B. passieren, wenn man eine Session über iSQL*Plus gestartet hat und der Browser dann abstürzt. Als Seiteneffekt kann es vorkommen, dass man dann bestimmte Tabellen, in denen zuvor Daten geändert wurden, in anderen Sessions nicht mehr bearbeiten kann (die Tabellen sind gelockt). Um alle Sessions gewaltsam zu beenden, kann man in SQL*Plus folgenden Befehl ausführen:

```
EXECUTE system.kill_my_sessions
```

Diese Hilfsprozedur beendet bis auf die Session, aus der man sie aufruft, alle anderen offenen Sessions eines Benutzers.

6.7 Help !

Oracle/SQL*Plus bietet eine kurze Online-Hilfe, die mit `help` aufgerufen wird, und sich dann selbst erklärt. Unter anderem erhält man mit `help commands` eine Liste aller Befehle von SQL*Plus.
Auf der Datenbankpraktikums-Homepage finden sich Links auf die wesentlichen weiteren Anleitungs-Dokumente: Für SQL-Befehle ist Oracle *SQL Reference* die erste Referenz, für allgemeine Datenbankkonzepte das Handbuch Oracle *Database Concepts*.

6.8 Mondial-Web-Service

Für Anfragen steht das aus der Datenbank-Vorlesung bekannte Web-Interface unter

`http://www.semwebtech.org/sqlfrontend/`

zur Verfügung. Dort kann man sich den auch intern optimierten Algebra-Baum anzeigen lassen.