

Kurzeinführung in Oracle

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

25. Oktober 2019

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.

Für jede Praktikumsgruppe existiert ein Verzeichnis

Gruppen-
verzeichnis

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

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.

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

Erreichbarkeit des CIP-Pools von aussen: mit

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

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

Die globalen Pfade für Oracle werden über ein Skript initialisiert. Dazu ist das folgende in die Datei `.bashrc` enthalten (falls nicht, reinschreiben):

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

Im einzelnen enthält dies (Stand April 2017) die folgenden Initialisierungen, die man ggf. bei einer eigenen Installation passend setzen muss:

```
# Oracle path in OpenAFS
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"
```

Jeder Teilnehmer muss seine lokale Umgebung entsprechend konfigurieren:

ORACLE_
WORK
setzen

- Die Datei `.bashrc` sollte die Zeile

```
export WORK=/afs/informatik.uni-goettingen.de/course/db-prakt/grpn
```

enthalten (wobei n die eigene Gruppennummer ist), um dann mit `cd $WORK` einfacher in dieses Verzeichnis wechseln zu können.

~/oracle/
erzeugen

- Weiterhin kann man noch ein Verzeichnis `oracle/` im eigenen Home-Verzeichnis erzeugen, in dem Skripte und Dateien für ORACLE abgelegt werden.

login.sql

- Die Bildschirmausgabe von ORACLE wird über ORACLE-Variablen gesteuert. Diese werden in der Datei `oracle/login.sql` gesetzt, die automatisch aufgerufen wird, wenn sqlplus gestartet wird. Um z.B. nach jeweils 50 Bildschirmzeilen auf eine Benutzereingabe zu warten, sollte diese Datei die folgenden Zeilen enthalten:

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

.bashrc/
global
anpassen

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 Variable `ORACLE_PATH` gibt an, wo SQL*Plus nach der Anlaufdatei “`login.sql`” sowie nach SQL-Skripten sucht. Es ist sinnvoll, `ORACLE_PATH` in `.bashrc` mit

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

zu setzen (in einer Zeile); in

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

befinden sich die Skripten zum Erstellen der Datenbasis.

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

```
export NLS_LANG=american_america.we8iso8859p1 für Englisch und
export NLS_LANG=german_germany.we8iso8859p1 für Deutsch.
```

- Um von SQL aus Skripte schreiben zu können, benötigt man einen Editor, der dann von SQL*Plus (dem SQL-Interface) aufgerufen wird. Der gewünschte Editor muss in der Umgebungsvariable `EDITOR` definiert werden (etwa `emacs` oder `xemacs` (sollte man können, braucht man immer wieder), `nano` (klein und handlich), oder `vi` (Geschmacksache)). Dies kann durch Editor ein-

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

in `.bashrc` erreicht werden. In SQL*Plus wird eine Datei durch `edit <filename>` in den Editor geladen.

Wenn man Veränderungen an diesen Files vorgenommen hat, sollte man sie mit “`source .bashrc`” neu ausführen 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 Das SQL-Interface SQL*Plus

SQL*Plus bietet ein interaktives Benutzerinterface (auch als SQL*Plus-Editor bezeichnet). Es gibt auch ein entsprechendes Web-Interface mit der gleichen Funktionalität und zusätzlichem Komfort (siehe Abschnitt 3.5).

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. SQL starten

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

```
passwd
```

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.

2.1 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.

2.2 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.

Skript
editieren

Editieren per SQL*Plus und eingestelltem Editor.

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.

Die Ausführung der Befehle erfolgt durch Eingabe von `start <filename>` (kürzer geht auch `@<filename>`).

Skript
ausführen

```
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 angegebenen 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.

3 How to get started – the first session

3.1 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. Vorausgesetzt, die Umgebungsvariable `ORACLE_PATH` enthält `/afs/informatik.uni-goettingen.de/group/dbis/public/Mondial`, geschieht dies in SQL*Plus wie folgt¹:

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:

¹Dies funktioniert nicht im Web-Interface iSQL*Plus!

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

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

3.2 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. Die Datenbank ist auf dem Stand von 1996.

3.3 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`.

3.4 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'
```

3.5 Web-Interfaces

Anfragen an globalen Account. 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.

Anfragen, Updates etc mit eigenem Account. Alternativ zu SQL*Plus, das in der Regel aus der Kommandozeile heraus gesteuert wird, kann man das Web-Interface iSQL*Plus benutzen, das die gleiche Funktionalität wie SQL*Plus bietet, dabei aber etwas komfortabler ist. Man kann iSQL*Plus mit jedem Browser, der Cookies akzeptiert, unter

```
https://ap34.ifi.informatik.uni-goettingen.de/isqlplus
```

erreichen. Dort wird man zur Angabe der Zugangsdaten aufgefordert, welche die gleichen wie für SQL*Plus sind. Als "Connect Identifier" gibt man `dbis` ein. Nach erfolgreichem Login erhält man ein Textfeld, in dem SQL-Anfragen eingegeben werden können. Die Ergebnisse einer Anfrage werden als Tabelle ausgegeben.

Hinweis: Die Skripte `create.sql` und `consult.sql` können aus diesem Interface nicht aufgerufen werden, da sie im Filesystem liegen.

3.6 Oracle SQL Developer

Im CIP-Pool der Informatik ist Oracles grafische Entwicklungsumgebung “SQL Developer” installiert. Dieses Tool ist eine Alternative zu SQL*Plus und dem Web-Interface mit vielen Funktionalitäten. In diese muss man sich natürlich ein wenig einarbeiten, dafür bekommt man solche Dinge wie:

- 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.

Aufruf:

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

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: oracle12c (bzw. oracle12c.informatik.uni-goettingen.de)
- Port: 1521
- SID: *nichts, wird seit Version 12 nicht mehr verwendet, stattdessen ...*
- Service-Name: dbis

(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.

3.7 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`.

3.8 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.

3.9 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*.