Kurzeinführung in Oracle

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

14. November 2003

1 Allgemeines

1.1 Administratives

Das Praktikum wird in der Regel in Gruppen zu 4 Personen durchgeführt. Die Teilnehmer loggen sich über `ssh` auf dem Rechner `dbpraktikum.ifi.informatik.uni-goettingen.de` ein.
Für das Arbeiten mit ORACLE ist zusätzlich zum **`dbpraktikum`-Account** des Instituts 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 zusätzlich auf dem Rechner `dbpraktikum` ein Directory `/home/db-prakt/gr`n, für das alle Gruppenmitglieder (sowie die Betreuer) Lese- und Schreibrecht besitzen. Dort sollen die SQL-Skripte der Gruppe abgelegt werden. Gruppen-directory
Zur Kommunikation stehen die Mail-Aliase

- `db-prak` : Betreuer und Teilnehmer
- `db-betr` : Betreuer

zur Verfügung. Folienkopien, Übungsblätter etc. werden auf
`http://dbis.informatik.uni-goettingen.de/Teaching/DBP/`
bereitgestellt.

1.2 Modifikationen an der UNIX-Umgebung

Jeder Teilnehmer muss seine lokale Umgebung entsprechend konfigurieren:

- Die Datei `.bash_profile` sollte die Zeile ORACLE_WORK setzen

  `export WORK=/home/db-prakt/gr`n

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

- Weiterhin sollte man noch ein Directory `oracle/` im eigenen Home-Directory erzeugen, in dem Dateien für ORACLE abgelegt werden. ~/oracle/ erzeugen

- Die Bildschirmausgabe von ORACLE wird über ORACLE-Variablen gesteuert. Diese werden in der Datei `oracle/login.sql` gesetzt. 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).

Sprache

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

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

- 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 `.bash_profile` mit

  ```
  export ORACLE_PATH=.:/home/db-prakt/Mondial:$HOME/oracle
  ```

  zu setzen; in `/home/db-prakt/Mondial` befinden sich die Skripten zum Erstellen der Datenbasis.

Editor einstellen

- 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 (Geschmackssache)). Dies kann durch

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

  in `.bash_profile` 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 .bash_profile` neu ausführen lassen.

Weitere Umgebungsvariablen Eine Beschreibung aller möglichen Umgebungsvariablen befindet sich in der "Oracle9i Administrator's Reference for UNIX Systems" (ein Link darauf findet sich auf der Praktikumsseite).
Eventuell von Interesse sind

ORACLE_HOME: Dies ist das Installations-Verzeichnis von ORACLE.

ORACLE_ADMIN: Legt das Verzeichnis für die installationsspezifischen SQL-Skripts fest (z.B. für die Definiton von Benutzerrollen oder die Erzeugung von ORACLE-Accounts). Hier befindet sich auch 'glogin.sql', die globale Anlaufdatei für SQL*Plus.

2 Das SQL-Interface SQL*Plus

SQL starten

SQL*Plus bietet ein interaktives Benutzerinterface (auch als SQL*Plus-Editor bezeichnet). SQL*Plus wird z.B. mit `sqlplus` in einer Shell aufgerufen. Man wird dann zur Eingabe des Benutzernamens (entspricht dem Account auf `dbpraktikum`) und des Passwortes aufgefordert, 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

```
ALTER USER <login> IDENTIFIED BY <new_password>;
```

geändert werden!

Alternativ kann man in der Datei *.bash_profile* einen alias definieren:

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

Nach einem `source .bash_profile` gengt der einfache Aufruf `sqlplus`, es muss nun nichts mehr eingegeben werden.

2.1 Interaktiv

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

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

SQLPlus 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` 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.

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 Unix-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 UNIX-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

Zuerst loggt man sich auf dem Rechner `dbpraktikum` 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 Account auf `dbpraktikum`) und des Passwortes aufgefordert, welches initial dem Benutzernamen entspricht. Nach einigen anderen Anzeigen erscheint als Prompt `SQL>`. Dann ändert man mit dem Befehl

```
ALTER USER <login> IDENTIFIED BY <new_password>;
```

das Passwort. Zu Beginn ist der ORACLE-Account noch leer, es muß erst die Datenbasis erzeugt werden. Vorrausgesetzt, die Umgebungsvariable `ORACLE_PATH` enthält den Pfad `/home/db-prakt/Mondial`, geschieht dies in SQL*Plus wie folgt: Datenbank erzeugen

```
@create.sql;
```

(Aufruf des SQL-Skripts `/home/db-prakt/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!

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 <table>;
```

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

```
SELECT * FROM <table>;
```

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 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.

3.4 Die tägliche Arbeit

Um die Aufgaben zu lösen, ist es sinnvoll, mit `cd $WORK` in das Gruppendirectory 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.

3.5 Web-Interface

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

`http://dbpraktikum.ifi.informatik.uni-goettingen.de/isqlplus`

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

Hinweis: Leider funktioniert das Aufrufen der Skripte `create.sql` und `consult.sql` bei diesem Interface nicht.

3.6 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.
Für SQL-Befehle ist ORACLE*9i SQL Reference* die erste Referenz, für allgemeine Datenbankkonzepte das Handbuch ORACLE*9i Database Concepts*. Auf der Datenbankpraktikums-Homepage finden sich mehrere nützliche Links.