Kurzeinführung in Oracle

Wolfgang May
may@informatik.uni-freiburg.de
Institut für Informatik,
Universität Freiburg

Oktober 2004

1 Allgemeines

1.1 Administratives

Das Praktikum wird in Gruppen zu drei bis vier Personen durchgeführt. Die Teilnehmer loggen sich über ihre eigenen UNIX-Accounts ein.
Für das Arbeiten mit ORACLE ist zusätzlich zum **Account auf den Unix-Workstations** 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.

Gruppen-directory

Für jede Praktikumsgruppe existiert zusätzlich ein Directory `/home/dbis/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 steht die Mailing-Liste

- `sql-user` : Betreuer und Teilnehmer

zur Verfügung. Folienkopien, Übungsblätter etc. werden auf

`http://dbis.informatik.uni-freiburg.de/index.php?course=SS06/Praktikum/Programmierung+in+SQL/index.html`

bereitgestellt.

1.2 Gruppenverzeichnis

Für jede Gruppe gibt es ein Verzeichnis
`/home/dbis/db-prakt/grp`x
wobei x die Gruppennummer ist. Zusätzlich wird jede Gruppe auch als UNIX-Gruppe `oragrp`x geführt.
Für das Tutorieren werden die SQL-Skripte einige Tage vor der Besprechung mit dem Tutor in das Gruppenverzeichnis gelegt. Hierfür legen Sie die Verzeichnisse `sheet1`–`sheet7` im Gruppenverzeichnis an und stellen den Code für die einzelnen Übungen in diese Verzeichnisse als `ex1.sql`, `ex2.sql`, `...` Der Code sollte

- richtig, insbesondere als SQL-Anfrage oder -Programm ausführbar sein,
- geeignet eingerückt sein und
- Kommentare enthalten, die ihn verstehbar machen.

Der Tutor kann so bereits vor der Besprechung auf die Programme schauen (und vielleicht Erweiterungen fordern).
Um das Gruppenverzeichnis in dieser Weise zu organisieren, werden die folgenden UNIX-Befehle benötigt:

- `cd /home/dbis/db-prakt/grp`*x*
- `mkdir sheet1` Das Verzeichnis wird erstellt, ist dann aber meist nur für Sie zugreifbar. Damit auch die Gruppenmitglieder das Verzeichnis nutzen können:
- `chmod 2770 sheet1`
- `chgrp oragrp`*x* `sheet1`
- `cd sheet1`
- Nun werde eine Datei `ex1.sql` erstellt, nutzbar durch die Gruppenmitglieder wird sie durch:
- `chmod 660 ex1.sql`
- `chgrp oragrp`*x* `ex1.sql`

Als Alternative können auch die Shell-Skripte `oracheck` und `orafix` benutzt werden, um die Zugriffsrechte zu kontrollieren und korrigieren. Die Skripte sind auf der Dokumenten-Praktikumsseite verfügbar.
Auch kann das Gruppenverzeichnis als CVS-Repositorium genutzt werden. Dann haben Sie Versionskontrolle und können zusätzlich E-Mail-Benachrichtigungen einrichten, wenn sich etwas ändert oder hinzugefügt wird.

1.3 Modifikationen an der UNIX-Umgebung

Jeder Teilnehmer muss seine lokale Umgebung entsprechend konfigurieren:

.bashrc/ login anpassen

- Die notwendigen Umgebungsvariablen für ORACLE werden durch `setup databases/oracle101` gesetzt. Dieser Befehl kann man entweder bei Bedarf von Hand aufrufen oder ihn in die Datei `.bashrc` (für die Shell bash) oder `.login` (für die tcsh[1]) schreiben.

  ORACLE_ WORK setzen

  Diese Datei sollte außerdem die Zeile

  `setenv WORK /home/dbis/db-prakt/grp`*n*

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

~/oracle/ erzeugen

- Weiterhin sollte man noch ein Directory `oracle/` im eigenen Home-Directory erzeugen, in dem 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. 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
  ```

  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-Meldungen ausgegeben werden, wird über die Umgebungsvariable `NLS_LANG` gesteuert.

[1] die Studentenaccounts laufen standardmäßig unter tcsh.

`setenv NLS_LANG american_america.we8iso8859p1` für Englisch und
`setenv 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 `.login` mit

  ```
  setenv ORACLE_PATH .:/home/dbis/db-prakt/MondialDB/oracle:$HOME/oracle
  ```

  zu setzen; in `/home/dbis/db-prakt/MondialDB/oracle` befinden sich die Skripte zum Erstellen der Datenbasis.

- 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), pico (klein und handlich), oder vi (Geschmackssache)). Dies kann durch

  Editor einstellen

  ```
  setenv EDITOR 'emacs -nw'
  ```

  in `.login` erreicht werden.

Werden diese Dateien geändert, sollte man sie mit `source <filename>` neu ausführen lassen.

Weitere Umgebungsvariablen Eine Beschreibung aller möglichen Umgebungsvariablen befindet sich im Anhang B des "Administrator's Reference Guide".
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 Definition 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*Plus bietet ein interaktives Benutzerinterface (auch als SQL*Plus-Editor bezeichnet). SQL*Plus wird z.B. mit `sqlplus` in einer Shell aufgerufen. Durch Angabe von Benutzername (des Oracle-Accounts!) und Kennwort wird eine Verbindung zu dem Oracle-Account hergestellt. Benutzername und Kennwort sind dann sinnvoll, wenn ein Benutzer mehrere Oracle-Accounts verwendet (also nicht beim Praktikum). Durch Angabe von `/` als Benutzername autorisiert man sich über den Unix-Account, es ist kein Kennwort erforderlich. In dieser Form kann das Einloggen auch kurz durch `sqlplus /` geschehen. Nach einigen anderen Anzeigen erscheint als Prompt `SQL>`. Mit `quit` wird SQL*Plus wieder verlassen.

SQL starten

SQL*Plus-Befehle sind Erweiterungen des Befehlsumfangs von SQL. Zusammen mit den bekannten SQL-Befehlen erlauben sie komfortablere Datenbankanfragen und -ausgaben. SQL- und SQL*Plus-Befehle können entweder interaktiv oder in Form von SQL-Skripts abgesetzt werden.

2.1 Interaktiv

Dabei werden die Befehle direkt hinter dem Prompt eingegeben. Interaktiv können immer nur einzelne SQL-Statements ausgeführt werden. Das SQL-Statement kann sich bei der Eingabe über mehrere Zeilen erstrecken. Die Zeilen werden laufend durchnummeriert und können so leichter editiert werden. Nach einem Strichpunkt am Zeilenende wird das SQL-Statement ausgeführt.

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

Anstatt die Kommandos 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.

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.

Editieren per SQL*Plus und eingestelltem Editor. Skript editieren
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>` oder kürzer durch `@<filename>`. Die Dateierweiterung `.sql` kann dabei weggelassen werden.

```
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. 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

Mit `sqlplus /` loggt man sich über das SQL-Interface auf dem Oracle-Account ein. Nach dem Erzeugen ist der Oracle-Account noch leer – d.h. ein leerer "Tablespace", in dem erst noch eine Datenbank erzeugt werden muss.
Die Mondial-Datenbasis wird dann durch den Aufruf von

Datenbank erzeugen

```
@create.sql;
```

aus SQL*Plus erzeugt (Aufruf des SQL-Skripts `/home/dbis/db-prakt/MondialDB/oracle/create.sql`). Dieses Skript ruft weitere Skripten auf, die die gesamte Datenbasis lokal erstellen. 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
`select * from city;`	sieht.
`select * from country;`	gibt diese Relation aus.
`delete from country;`	löscht die Relation, was dann durch
`commit;`	endgültige 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, pico usw. aufzurufen. Innerhalb der Gruppe sollte man sich über die Namensgebung der SQL-Skripts einig sein.

3.5 Hilfe!

Oracle/SQL*Plus bietet eine kurze Online-Hilfe, die mit `help` auf gerufen wird, und sich dann selbst erklärt. Unter anderem erhält man mit `help commands` eine Liste aller Befehle von SQL*Plus, SQL und PL/SQL.
Für SQL-Befehle ist Oracle *Server SQL Language* die erste Referenz, für allgemeine Datenbankkonzepte das Oracle *Server Concepts Manual*. Auf der Datenbankpraktikums-Homepage ist ein klickbares Manual als Link zu finden.