Entwicklungsumgebung für die Laborübung

VL Datenbanksysteme

Ingo Feinerer

Arbeitsbereich Datenbanken und Artificial Intelligence
Institut für Informationssysteme
Technische Universität Wien

Gliederung

Übung

- Adresse: `bordo.dbai.tuwien.ac.at`
- Zugriff:
  - Von „überall“ her möglich (z.B. Übungsräume, andere Rechner im TU-Netz, zu Hause, in der Firma, etc.)
  - ausschließlich mittels **Secure Shell** (egal von wo aus)
- Betriebssystem auf bordo: SUSE Linux
- Linux Kennung:
  - Alle Übungsteilnehmer erhalten nach der Anmeldung eine Linux Kennung mit Usernamen `u<matrikelnummer>`.
  - Das Passwort entspricht dem Passwort im LVA-Betreuungssystem.
  - Passwort ändern: mit Linux-Befehl **passwd**. Geändertes Passwort nicht vergessen!

Secure Shell

Von Windows-PC aus:

- SSH-Client (PuTTY) und SCP-Client (PSCP): `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`
- Liste von Alternativen: `http://www.jfitz.com/tips/ssh_for_windows.html`

Von Linux-PC aus:

- ssh und scp sind ohnehin vorhanden
- Bequemer Datentransfer: sftp mit Konquerer oder Nautilus

Einloggen auf Bordo bzw. File Transfer:

Beispiele

- ssh username@bordo.dbai.tuwien.ac.at
- scp bsp1.sql username@bordo.dbai.tuwien.ac.at:abgabe1.sql

PostgreSQL Datenbank auf Bordo

DB-Server auf Bordo:

- PostgreSQL 8.4

DB-Benutzung:

- Alle LVA-Teilnehmer erhalten eine eigene DB-Kennung in der Datenbank
- DB-Username: gleich wie Linux-Username
- Starten einer interaktiven Session: mit dem Kommando psql (im ssh-Fenster). Login läuft über SSH Account, kein Username/Passwort notwendig
- Für remote connections: DB-Passwort ändern mit \password in psql

Editoren auf Bordo

- **emacs**: Hilfe im Web, z.B. `http://www.gnu.org/software/emacs/tour/`
- **vi**: Hilfe im Web, z.B. `http://www.library.yale.edu/wsg/docs/vi_hands_on/`
- **pico**: Hilfe im Web, z.B. `http://www.uic.edu/depts/accc/software/pine/pico.html`
- Alternativen: Entwicklung von SQL und PL/SQL auf dem lokalen PC und Upload nach Bordo mittels scp (oder sftp).

Gliederung

Datei Befehle

- Wichtige Befehle für die Laborübung

`\o datei`	Output wird in `datei` geschrieben
`\o`	Ende der Ausgabe in `datei`
`\i datei`	Führt die Befehle aus `datei` aus

Beschreibungsbefehle

- `\d+` Tabellen auflisten
- `\df+` Funktionen auflisten
- `\d+ table` Tabelle beschreiben
- `\df+ functions` Funktion beschreiben

Beispiel

```
SELECT column_name FROM
   INFORMATION_SCHEMA.COLUMNS
   WHERE table_name = 'table';
```

pgAdmin

- Grafisches Tool zur Datenbankadministration
- SQL Editor, Anzeigen von Tabellen, Datenbanken
- In der Laborübung:
  - SSH-Port Forwarding von Port 5432
  - Username: `u<Matrikelnummer>`
  - Passwort: vorher mit \password in psql festlegen
- `http://www.pgadmin.org`

Gliederung

SQL Statement

Beispiel

Suche die Namen jener Studenten, die alle 4-stündigen Vorlesungen gehört haben.

- Ersetzung von „for all“ durch „exists“:
  Suche die Namen jener Studenten, für die *nicht* gilt: Es gibt eine 4-stündige VO, für die *nicht* gilt: der Student hat diese VO gehört.
- Geschachteltes SQL-Statement erstellen: „schrittweise“ von innen nach außen.

SQL Statement

Innerstes **SELECT**:

- Wähle MatrNr eines beliebigen Studenten sowie Vorlesungsnummer einer beliebigen Vorlesung:

```
SELECT * FROM hoeren ORDER BY VorlNr
SELECT MatrNr FROM studenten
SELECT DISTINCT MatrNr FROM hoeren
```

  z.B., wähle MatrNr 26120 und VorlNr 5001

- **SELECT** Statement (in Datei test1.sql editieren):

```
SELECT * FROM hoeren h
WHERE h.VorlNr = 5001 AND h.MatrNr = 26120
```

- Datei einlesen und ausführen in psql: `\i test1.sql`

SQL Statement

Mittleres **SELECT**:

- Gesucht: Informationen über alle 4-stündigen Vorlesungen, die von einem bestimmten Studenten (z.B. wieder der Student mit MatrNr 26120) nicht gehört wurden.
- **SELECT** Statement (in Datei test2.sql editieren):

```
SELECT * FROM Vorlesungen v
WHERE v.SWS = 4 AND NOT EXISTS
   (SELECT * FROM hoeren h
    WHERE h.VorlNr = v.VorlNr
    AND h.MatrNr = 26120)
```

- Datei einlesen und ausführen in psql: `\i test2.sql`

SQL Statement

Äußeres **SELECT**:

- Gesucht: Informationen über Studenten, für die es keine 4-stündigen Vorlesungen gibt, die von diesen Studenten nicht gehört wurden.
- **SELECT** Statement (in Datei abgabe_bsp1.sql editieren):

```
SELECT * FROM Studenten s
WHERE NOT EXISTS
   (SELECT * FROM Vorlesungen v
    WHERE v.SWS = 4 AND NOT EXISTS
       (SELECT * FROM hoeren h
        WHERE h.VorlNr = v.VorlNr
        AND h.MatrNr = s.MatrNr))
```

- Datei einlesen und ausführen in SQL*Plus:
  \i abgabe_bsp1.sql

SQL Statement

Bemerkung:
Durch **INSERT**s und **DELETE**s die Testdaten so ändern, dass die Anfrage zumindest 2 bis 3 Tupel liefert.

```
INSERT INTO hoeren VALUES (24002,4052);
INSERT INTO hoeren VALUES (24002,4630);
INSERT INTO hoeren VALUES (24002,5001);
INSERT INTO hoeren VALUES (24002,5041);
```

PL/SQL Prozedur

- Gesucht: Prozedur, die den Namen und die Semesteranzahl eines Studenten ausgibt
- Mögliche Vorgangsweise:
  - Entwicklung mittels „normalem“ Editor
  - Am Anfang: nur Prozedur-Kopf und „NULL“-Body
  - „Schrittweise“ den Body hinzufügen
  - Nach jedem Schritt kompilieren.
  - „Test-Treiber“ erstellen
  - Bildschirmausgaben mittels RAISE NOTICE / RAISE DEBUG zwecks Fehlersuche

PL/SQL Prozedur

- Datei versuch1.sql:

```
CREATE OR REPLACE FUNCTION suche(matrnr numeric(10))
   RETURNS void AS $$
DECLARE
BEGIN
   NULL;
$$ LANGUAGE plpgsql;

SELECT suche(0620611);
```

- Einlesen und Kompilieren: \i versuch1.sql

PL/SQL Prozedur

- Datei versuch2.sql:

```
CREATE OR REPLACE FUNCTION suche(matrnr numeric(10))
  RETURNS void AS $$
DECLARE
name      varchar(30);
semester  numeric(2);
BEGIN
  SELECT s.name, s.semester INTO name, semester
    FROM students s WHERE s.matrnr = matrnr;
    RAISE NOTICE 'Name: %, Semester: %', name, semester;
$$ LANGUAGE plpgsql;

SELECT suche(0620611);
```

- Einlesen und Kompilieren: \i versuch2.sql
- Aber was wenn kein Ergebnis gefunden wurde?

PL/SQL Prozedur

- Datei versuch3.sql:

```
CREATE OR REPLACE FUNCTION suche(matrnr numeric(10))
  RETURNS void AS $$
DECLARE
name      varchar(30);
semester  numeric(2);
BEGIN
  SELECT s.name, s.semester INTO name, semester
    FROM students s WHERE s.matrnr = matrnr;
  IF (name IS NULL) THEN
    RAISE NOTICE 'Leider nichts gefunden';
  ELSE
    RAISE NOTICE 'Name: %, Semester: %', name, semester;
  END IF;
$$ LANGUAGE plpgsql;

SELECT suche(0620611);
```

- Einlesen und Kompilieren: \i versuch3.sql