Übung DP: Datenbankprogrammierung

1. Selbststudium

Frage 1

Was ist ein Cursor? Definieren Sie das Konzept in Ihren eigenen Worten.

- Ein Cursor ist ein Zeiger, der eine Reihe von Tupeln in einer bestimmten Reihenfolge (die physische oder eine durch `ORDER BY` definierte) durchlaufen kann.
- Ein Cursor wird mittels `DECLARE`-Anweisung erstellt für ein `SELECT`-Statement erzeugt:
  - `DECLARE foo CURSOR FOR SELECT * FROM person ORDER BY AGE DESC;`
- Mittels `OPEN` und `CLOSE` kann ein Cursor geöffnet bzw. geschlossen werden:
  - `OPEN foo`
  - `CLOSE foo`
- Der `FETCH`-Befehl liefert die Tupel zurück, auf die der Cursor gegenwärtig verweist, und positioniert den Cursor um eine Tupel weiter.

Frage 2

Aus welchem Grund (warum) und zu welchem Zweck (wozu) braucht man Cursors?

- Warum: Die Ergebnisse vom `SELECT`-Statement sind oft zu gross, um sie auf einmal an das anfragende Programm zu übertragen.
- Wozu: Mit einem Cursor kann die Ergebnismenge eines `SELECT`-Statements in mehreren Schritten zum anfragenden Programm übertragen werden, denn die Datenbank weiss "weiss" anhand des Cursors nach der Übertragung eines Teilergebnisses, an welcher Stelle er mit der Übertragung fortgefahren werden muss.

Frage 3

Wozu werden Datenbanksprachen in andere Sprachen eingebettet?

- Benutzer, die nicht mit SQL umgehen können, brauchen eine spezielle Anwendungssoftware. Anwendungssoftware wird zumeist nicht in SQL, sondern mit einer anderen Programmiersprache geschrieben. Diese andere Programmiersprache muss eine Möglichkeit haben, SQL-Befehle auf eine Datenbank abzusetzen.

2. Hello JDBC

```
package main;

import java.sql.Connection;
```

```java
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

public class HelloJDBC {

    public static void main(String[] args) {
        final String connectionString = "jdbc:postgresql://127.0.0.1:5432/unidb";
        try {
            Class.forName("org.postgresql.Driver");
            Connection connection = DriverManager.getConnection(connectionString,
                "postgres", "postgres");
            Statement statement = connection.createStatement();
            ResultSet professoren = statement.executeQuery("SELECT persnr FROM professoren");
            Map<Integer, List<String>> vorlesungenByProfessor = new TreeMap<>();
            while (professoren.next()) {
                int persnr = professoren.getInt("persnr");
                vorlesungenByProfessor.put(persnr, new ArrayList<>());
                PreparedStatement prep = connection.prepareStatement(
                    "SELECT titel FROM vorlesungen WHERE gelesenvon = ?");
                prep.setInt(1, persnr);
                ResultSet vorlesungen = prep.executeQuery();
                while (vorlesungen.next()) {
                    vorlesungenByProfessor.get(persnr).add(vorlesungen.getString("titel"));
                }
            }
            for (Integer persnr : vorlesungenByProfessor.keySet()) {
                System.out.println("Professor mit PersNr. " + persnr + " liest:");
                for (String vorlesung : vorlesungenByProfessor.get(persnr)) {
                    System.out.println(vorlesung);
                }
                System.out.println();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Ausgabe:

```
Professor mit PersNr. 2125 liest:
Ethik
Maeeutik
Logik

Professor mit PersNr. 2126 liest:
Erkenntnistheorie
Wissenschaftstheorie
Bioethik

Professor mit PersNr. 2127 liest:

Professor mit PersNr. 2133 liest:
Der Wiener Kreis

Professor mit PersNr. 2134 liest:
Glaube und Wissen

Professor mit PersNr. 2136 liest:

Professor mit PersNr. 2137 liest:
Grundzuege
Die 3 Kritiken
```

3. Stored Procedures

```
create table t (datum date primary key, yay boolean);

create or replace function f (startdate date, anz int)
returns void
as $$
declare
    i int;
    nextDay date;
begin
    i := 1;
    nextDay := startdate;
    while (anz >= i) loop
        if (extract(dow from nextDay) in (6, 0)) then
            insert into t (datum, yay)
            values (nextDay, true);
```

```
        else
            insert into t (datum, yay)
            values (nextDay, false);
        end if;
        i := i + 1;
        nextDAy := nextDay + INTERVAL '1 day';
    end loop;
end;
$$
language 'plpgsql';

select f('2016-11-11', 111);

select * from t;
```

Ausgabe:

```
datum         yay
-----         ---
2016-11-11    f
2016-11-12    t
2016-11-13    t
...
2017-02-27    f
2017-02-28    f
2017-03-01    f
```

Welche Semantik hat die Prozedur `f` (was genau macht das Programm)? Was bedeuten die Daten, welche die Prozedur generiert?

- Das Program ermittelt von einem Startdatum (Parameter `startdate`) an für eine Anzahl von Tagen (Parameter `anz`), ob der jeweilige Tag auf ein Wochenende fällt, sprich `dow` (day of week) 6 (Samstag) oder 0 (Sonntag) zurückgibt. Ist dies der Fall, wird das jeweilige Datum mit `yay = true` in die Tabelle `t` geschrieben, sonst wird der neue Eintrag mit `yay = false` geschrieben. Das Programm fährt mit dem nächsten Tag fort, bis die gewünschze Anzahl Tage abgearbeitet sind.

4. Verbindung von JDBC mit Stored Procedures

```
package main;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

```java
import java.sql.Statement;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class Exercise4 {

    public static void main(String[] args) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        LocalDate fromDate = LocalDate.parse(args[0], formatter);
        int numberOfDates = Integer.valueOf(args[1]);
        final String connectionString = "jdbc:postgresql://127.0.0.1:5432/unidb";
        try {
            Class.forName("org.postgresql.Driver");
            Connection connection = DriverManager.getConnection(connectionString,
                "postgres", "postgres");
            Statement statement = connection.createStatement();
            statement.execute("DELETE FROM t");
            PreparedStatement prep = connection.prepareStatement("SELECT f(?, ?)");
            prep.setDate(1, java.sql.Date.valueOf(fromDate));
            prep.setInt(2, numberOfDates);
            prep.execute();
            ResultSet days = statement.executeQuery("SELECT datum, yay FROM t");
            System.out.println("datum\t\tyay");
            System.out.println("-----\t\t---");
            while (days.next()) {
                System.out.println(days.getString("datum") + "\t" +
                    days.getBoolean("yay"));
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Ausgabe:

```
datum       yay
-----       ---
2017-10-01  true
2017-10-02  false
2017-10-03  false
2017-10-04  false
2017-10-05  false
2017-10-06  false
2017-10-07  true
```

```
2017-10-08  true
2017-10-09  false
2017-10-10  false
```

5. Stored Functions und Cursors

```
create or replace function median_semester()
returns float
as $$
declare
    stud_cursor cursor for select semester from studenten order by semester asc;
    size int;
    a int;
    b int;
    sem int;
    i int;
    lower int;
    upper int;
begin
    size := 0;
    i := 1;
    open stud_cursor;
    loop
        fetch stud_cursor into sem;
        exit when not found;
        size = size + 1;
    end loop;
    close stud_cursor;
    /* even number of entries: calculate mean of middle two */
    if size % 2 = 0 then
        a := size / 2;
        b := a + 1;
        open stud_cursor;
        loop
            fetch stud_cursor into sem;
            if i = a then
                lower = sem;
            end if;
            if i = b then
                upper = sem;
                exit;
            end if;
            i := i + 1;
```

```
        end loop;
        close stud_cursor;
        return (lower + upper) / 2;
    /* odd number of entries: return middle entry */
    else
        a = size / 2;
        open stud_cursor;
        loop
            fetch stud_cursor into sem;
            if i = a then
                lower = sem;
                exit;
            end if;
            i := i + 1;
        end loop;
        close stud_cursor;
        return lower;
    end if;
end;
$$ language 'plpgsql';

select median_semester();
```