6. Versuch: SQL und Java

Setzen Sie für diesen Versuch einige Umgebungsvariablen:

```
source /afs/informatik.uni-goettingen.de/group/dbis/public/oracle/.oracle_env
# enthaelt so in etwa das folgende:
export ORACLE=/afs/informatik.uni-goettingen.de/group/dbis/public/oracle
export ORACLE_HOME=$ORACLE/instantclient
export CLASSPATH=.:$ORACLE/sqlj/lib/runtime12.jar
                  :$ORACLE/sqlj/lib/translator.jar
                  :$ORACLE_HOME/ojdbc8.jar
                  :$CLASSPATH
export $PATH=$ORACLE_HOME:$ORACLE/bin:$ORACLE/sqlj/bin:$PATH
```

Die Java-URL des Oracle ist

```
jdbc:oracle:thin:@//oracle19c.informatik.uni-goettingen.de:1521/dbis.informatik.uni-goettingen.de
```

Bei JDBC und SQLJ wird auf externe Dateien zugegriffen, in der die Login-Daten für Oracle in Klartext gespeichert sind. Im Verzeichnis `$WORK` sind diese Dateien von allen Gruppenmitgliedern lesbar. Man sollte also in Erwägung ziehen, die Klassen im Home-Verzeichnis zu entwickeln.

Aufgabe 6.1 (Java Stored Procedures, Konsistenzerhaltung; 20 P.)

(Benutzen Sie für diese Aufgabe Mondial mit den auf Blatt 4 erstellten referentiellen Integritätsbedingungen, aber ohne die auf Blatt 5 erstellten Trigger)

Überlegen Sie sich, welche Aktionen bei Löschung eines Flusses/Sees/Meeres in der Datenbank sinnvoll sind. Schreiben Sie eine Java-Klasse mit drei Methoden `deleteRiver/Lake/Sea`, die ein(en) gegebenes/n Fluss/See/Meer als Parameter erhalten, löschen und alle relevanten Tabellen in der Datenbank entsprechend aktualisieren. Laden Sie die Klasse in die Datenbank und erzeugen Sie freie Prozeduren als Wrapper.

Experimentieren Sie etwas damit um das Verhalten Ihrer Procedure zu validieren.
Vielleicht: Fällt Ihnen dabei irgendetwas auf?

Aufgabe 6.2 (JDBC: Einbindung in eine grafische Benutzeroberfläche; 30 P.)

Erstellen Sie eine einfache grafische Benutzeroberfläche, mit der Anfragen *und Updates* an Ihre SQL-Datenbank übermittelt werden können. Benutzen Sie dazu eine beliebige Java-Bibliothek (z.B. AWT, Swing, JavaFX, oder SWT).

Die grafische Benutzeroberfläche soll folgendes enthalten:

- Ein Textfeld zur Eingabe des Statements (Anfrage oder sonstiges DML-Statement, d.h INSERT, DELETE, oder UPDATE),
- Einen Button zum Absenden,
- Ein Feld, in dem die Ausgabe dargestellt wird. Passend zum ausgeführten SQL-Befehl soll die Ergebnismenge oder die Anzahl der geänderten Zeilen ausgegeben werden. Beachten Sie, dass die Anzahl der Spalten der Ergebnismenge je nach Anfrage variieren kann. Vor den Ergebnistupeln soll eine Zeile mit den Attributnamen der betreffenden Spalte erscheinen.

Aufgabe 6.3 (SQLJ, Bruttoinlandsprodukt; 10 P.)
Berechnen Sie mit Hilfe von SQLJ (und ohne Hilfstabellen) analog zu Aufgabe 4.2, welcher Anteil der Weltbevölkerung mindestens notwendig ist, um 50% des Welt-Bruttoinlandsproduktes zu erzeugen. Nehmen Sie dabei an, dass sich das Bruttoinlandsprodukt eines Landes gleichmäßig auf alle Einwohner verteilt. Geben Sie an, wieviele Einwohner aus welchen Ländern beteiligt sind. Verwenden Sie einen Iterator über ein geeignet gewähltes `SELECT`-Statement.

Aufgabe 6.4 (Visualisierung: Bevölkerungsentwicklung USA, 10 P.)
Visualisieren Sie die Veränderung der Einwohnerzahlen der einzelnen Städte der USA zwischen 1980 und 2020, wie folgt:

- Rechteckiges Feld entsprechend den Breiten- und Längengeraden,
- Markieren der Städte durch farbige Kreise entsprechend dem Bevölkerungswachstum bzw. Rückgang, im Bereich grün-gelb-rot.

Verwenden Sie dazu ein JDBC oder SQLJ sowie ein Java-Paket, das Grafiken/Fenster unterstützt (z.B. Swing, JavaFX oder SWT).

Wenn Sie möchten, können Sie auch eine beliebige andere Programmier- oder Visualisierungssprache benutzen (z.B. "R") – die Verbindung zur Datenbank funktioniert eigentlich immer auf dieselbe Art.

Was fällt auf?

Aufgabe 6.5 (Graphenalgorithmus mit JDBC; 20 P.)
In Aufgabe 4.5 wurde ein Graphenalgorithmus in PL/SQL implementiert. Lösen Sie dieselbe Aufgabe mit Java+JDBC (verwenden Sie für mehrfach verwendete Anfragen und Updates PreparedStatements). Vergleichen Sie die Laufzeit.

Aufgabe 6.6 (Bruttoinlandsprodukt; 10 P.)
Lösen Sie Aufgabe 4.2 ohne Verwendung von JDBC/SQLJ oder PL/SQL durch eine normale `SELECT-FROM-WHERE`-Anfrage (ohne Verwendung von LIMIT, ROWNUM etc.). Definieren Sie ggf. geeignete Views für Zwischenergebnisse.

Hier ist offensichtlich die Nuss zu knacken, wie das in SQL überhaupt deskriptiv geht.

Hinweis: Geben Sie im ersten Schritt nur das Land aus, das am Ende nur teilweise beiträgt, sowie sie Zahl der benötigten Personen.

Vergleichen Sie deren Komplexität mit der Lösung in PL/SQL bzw. JDBC bzw. SQLJ.

Abgabe bis 12.7.2024