Aufgabenblatt 6

— Java Database Connectivity (JDBC) —

Ausgabe am 14.01.2008
Abgabe bis 23.01.2008, 13.30 Uhr

Wichtige Bemerkungen:

- Beachten Sie bitte, dass die Lösungen dieser Übungsserie *ausschließlich* per E-Mail abzugeben sind. Schicken Sie uns bitte sowohl die Quellcode-Datei (`*.java`) als auch die Java Bytecode-Datei (`*.class`) an `dbs1-2007@hpi.uni-potsdam.de` mit dem Betreff „Abgabe DBS I: Aufgabenblatt 6 [Namen | Matrikelnummern]“.
- Beachten Sie ferner, dass die Bearbeitung der Aufgaben ausschließlich in Zweiergruppen erfolgen soll, d. h. also insbesondere, dass die Abgabe von identischen oder zu ähnlichen Lösungen *nicht gestattet* ist.
- Für die erfolgreiche Bearbeitung dieser Übungsserie ist es erforderlich, die IMDB-Daten erneut einzuladen (vgl. Aufgabe 4 des zweiten Übungsblatts). Gehen Sie dazu wie folgt vor: Löschen Sie mittels `DELETE FROM ACTOR` sämtliche Tupel der Tabelle `ACTOR` (analog für die vier anderen Tabellen). Importieren Sie anschließend die entsprechenden Daten erneut, wobei Sie hierfür die UTF8-kodierten Dateien aus dem Verzeichnis `~dbs1u00/imdbUTF8` (auf dem Server `isis.hpi.uni-potsdam.de`) verwenden. Überprüfen Sie anschließend den fehlerfreien Import, indem Sie die entsprechenden Logging-Dateien auf mögliche Fehlermeldungen hin untersuchen.
- Sie können auf den Server `isis.hpi.uni-potsdam.de` nur innerhalb des HPI-Netzes zugreifen. Soll Ihr Programm von einem anderen Rechner aus eine Verbindung zu diesem Server herstellen, ist der Umweg über einen „von außen“ sichtbaren HPI-Rechner (z. B. `placebo.hpi.uni-potsdam.de`) notwendig. Wollen Sie z. B. auf den Server über den lokalen Port 50000 zugreifen, so rufen Sie

  ```
  ssh -L 50000:isis:50050 dbs1uXX@placebo.hpi.uni-potsdam.de
  ```

  auf (ersetzen Sie `XX` entsprechend). Nun können Sie über `localhost:50000` auf das DBMS und Ihre Datenbank zugreifen.
- Der DB2-JDBC Treiber (`db2jcc.jar` für JDBC Version ≤ 3.0 bzw. `db2jcc4.jar` für JDBC Version 4.0) liegt im Verzeichnis `~dbs1u00/jdbc` (auf dem Server `isis.hpi.uni-potsdam.de`).
- Eine gute Hilfe bieten u. a. das Manual „Developing Java Applications“, das Sie unter `http://www-1.ibm.com/support/docview.wss?rs=71&uid=swg27009552` herunterladen können.

Aufgabe 1: Konsolidierung der IMDB-Datenbank **55 P**

In den vergangenen Übungen wurden bereits einige Unzulänglichkeiten unseres Ausschnittes der IMDB-Datenbank deutlich (z. B. SchauspielerInnen, die in Filmen mitspielen, die nicht in der Tabelle `MOVIE` enthalten sind). Das Ziel dieser Aufgabe besteht unter anderem darin, solche Probleme zu beheben.

Schreiben Sie unter Verwendung von JDBC *ein* Java-Programm, das sämtliche nachfolgend aufgeführten Aktionen nacheinander ausführt. Das Programm soll so geschrieben sein, dass es mit dem Benutzernamen und zugehörigem Passwort als Argumente aufgerufen werden kann, sich unter Nutzung dieser Informationen automatisch zur IMDB-Datenbank (DB-Name: `dbs1imdb`, Server: `isis.hpi.uni-potsdam.de`, Port: `50050`) verbindet und die Verbindung am Ende selbständig wieder schließt. Es soll keinerlei Benutzerinteraktion geben. Überlegen Sie in jeder Teilaufgabe, ob der Einsatz eines `PreparedStatement`-Objektes sinnvoll ist und verwenden Sie dieses gegebenenfalls. Beachten Sie, dass das Programm so zu schreiben ist, dass es unter jedem möglichen Zustand der IMDB-Datenbank (und nicht nur in unserem Beispielzustand) korrekt arbeitet.

(a) Erstellen Sie eine Tabelle `ACT`, die wie folgt aufgebaut ist und zur Speicherung von Schauspielern und Schauspielerinnen dienen soll.

Tabellenname: `ACT`

Spaltenname	Datentyp	Nullwert zugelassen?	Primärschlüsselattribute
`Firstname`	varchar(255)	nein	×
`Lastname`	varchar(255)	nein	×
`isFemale`	smallint	nein	–

Geben Sie zusätzlich die Bedingung an, dass `isFemale` nur die Werte 0 (für den Wahrheitswert falsch) und 1 (für den Wahrheitswert wahr) annehmen kann.[1] **3 P**

(b) Speichern Sie jeden Schauspieler bzw. jede Schauspielerin, der bzw. die in der Tabelle `ACTOR` bzw. `ACTRESS` gespeichert ist *und* in mindestens einem in der Tabelle `MOVIE` gespeicherten Film mitspielt, in der neu angelegten Tabelle `ACT`.

Lesen Sie dazu die entsprechenden Namen aus `ACTOR` und `ACTRESS` aus und zerlegen Sie diese in Vor- und Nachnamen, so dass Sie diese anschließend in der Tabelle `ACT` speichern können. Behandeln Sie die Namen Adrian (V) und Janus als Nachnamen und speichern Sie für den Vornamen jeweils die leere Zeichenkette. **7 P**

(c) Zwischen dem Entitätstyp Film und dem Entitätstyp SchauspielerIn besteht eine n:m-Beziehung: ein/eine Schauspieler/-in kann in mehreren Filmen mitspielen; in einem Film können mehrere SchauspielerInnen mitspielen. Ein solcher Relationship-Typ wird bei der Übersetzung ins relationale Modell auf eine Relation abgebildet, die sämtliche Primärschlüsselattribute der Tabellen `MOVIE` und `ACT` als zusammengesetzten Primärschlüssel besitzt. Darüber hinaus bilden eventuell vorhandene Attribute des Relationship-Typs die Nichtschlüsselattribute der entsprechenden Relation.

[1] Beachten Sie, dass DB2 den Datentyp `BOOLEAN` nicht unterstützt.

Legen Sie eine entsprechende Tabelle `ACTMOVIE`, die zur Speicherung dieser Relation dient, an. Definieren Sie entsprechende Fremdschlüssel, die sichern, dass Tupel in dieser Tabelle ausschließlich Filme in `MOVIE` und SchauspielerInnen in `ACT` referenzieren. Geben Sie bei der Definition der Fremdschlüssel `ON DELETE CASCADE` an.

Beachten Sie, dass die Attribute (`Role`, `Order`) des Relationship-Typs in der vorliegenden Übersetzung als Attribute in der Tabelle `ACTOR` (bzw. `ACTRESS`) vorliegen. Beachten Sie bei der Definition ferner, dass beide Attribute optional sind. **5 P**

(d) Befüllen Sie die angelegte Tabelle `ACTMOVIE` ausgehend von den Daten, die in den Tabellen `ACTOR` bzw. `ACTRESS` gespeichert sind. Offensichtlich nicht vorhandene Werte für die Attribute `Role` und `Order` sollen jeweils durch den `NULL`-Wert repräsentiert werden. **7 P**

(e) Löschen Sie nun diejenigen Tupel aus den Tabellen `ACTOR` und `ACTRESS`, die sich aus den in `ACTMOVIE` und `ACT` gespeicherten Daten ergeben. Damit verbleiben in `ACTOR` bzw. `ACTRESS` nur noch solche Tupel, die auf einen Film verweisen, der nicht in `MOVIE` existiert. Setzen Sie für diese Tupel den Wert des Attributs `Movie_ID` auf den `NULL`-Wert. Beachten Sie, dass es dafür evtl. zuvor erforderlich ist, die Definition der Tabelle `ACTOR` bzw. `ACTRESS` entsprechend zu ändern. **5 P**

(f) Geben Sie die durchschnittliche Länge der Vor- bzw. Nachnamen für die in `ACT` gespeicherten SchauspielerInnen aus. **3 P**

(g) Schreiben Sie eine Methode `void getArtists(String firstname, String lastname)`, die die Namen aller SchauspielerInnen ausgibt, die in einem Film mit dem/der durch die Eingabeparameter angegebenen Schauspieler/in zusammengearbeitet haben. Achten Sie dabei darauf, dass Sie eine/n Schauspieler/in nicht mehrfach ausgeben. Die Ausgabe soll dabei nach den Vornamen geordnet erfolgen.

Geben Sie nun unter Verwendung der Methode `getArtists` die ehemaligen KollegInnen von Wolf Roth bzw. von Cecile Nordegg aus. **7 P**

(h) Schreiben Sie eine Methode `void getTatortMovies()`, die den Titel und das jeweilige Alter (in Jahren) der in der Tabelle `MOVIE` gespeicherten Filme der Kriminalserie „Tatort“ ausgibt. Die Ausgabe soll nach dem Alter absteigend sortiert sein. Bei gleichem Alter soll nach dem Titel aufsteigend sortiert werden. **5 P**

(i) Schreiben Sie eine Methode `void getChain(char c)`, die ausgehend von dem Eingabezeichen `c` die *längste* Kette $(n_1, \ldots, n_k)$ von in `ACT` gespeicherten Nachnamen ausgibt, so dass

- n_1 mit `c` beginnt,
- ein Nachname nicht mehrfach in dieser Kette auftritt und
- der letzte Buchstabe von n_i mit dem ersten Buchstaben von n_{i+1} übereinstimmt (für $i = 1, \ldots, k-1$).

Gibt es für ein `c` mehrere solcher Ketten mit maximaler Länge, so geben Sie alle diese Ketten aus. **13 P**