Übungen zu Kapitel 24

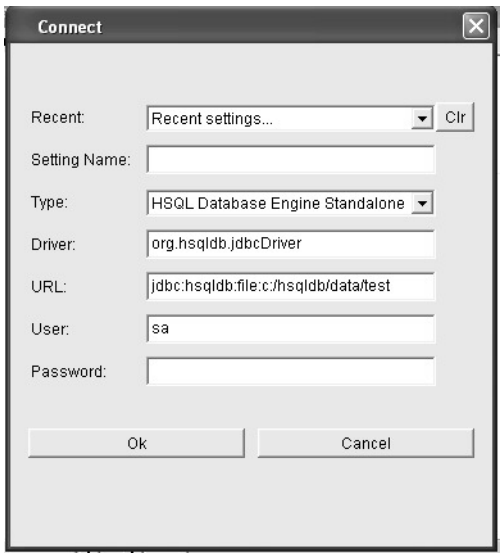
24.1 HSQLDB DatabaseManager

Mit dem Datenbanksystem HSQLDB steht ein Database-Management Tool zur Verfügung, das man mit folgendem Kommando (durch Eingabe in der Konsole) startet:

```
java -cp <hsqldb-Verzeichnis>/lib/hsqldb.jar  org.hsqldb.util.DatabaseManager
```

Der Ausdruck *<hsqldb-Verzeichnis>* ist natürlich entsprechend anzupassen. Im ***Connect***-Fenster sind die folgenden Einträge vorzunehmen bzw. auszuwählen:

- Type: ***HSQL Database Engine Standalone***
- Driver: ***org.hsqldb.jdbcDriver*** (wird automatisch eingestellt)
- URL: ***jdbc:hsqldb:file:<Db-Daten-Verzeichnis>/<Db-Name>***
  Hierbei bezeichnet ***<Db-Daten-Verzeichnis>*** das Verzeichnis, in dem die Datenbank-Dateien abgelegt werden (`c:/hsqldb/data`) und ***<Db-Name>*** den Namen der Datenbank-Dateien (`test`). Diese Namen sind grundsätzlich frei wählbar, das Verzeichnis sollte jedoch existieren.



- User: ***sa*** (wird automatisch eingestellt)
- Password: (keine Eingabe)

Nach Betätigen des "OK"-Buttons startet der Manager die Datenbank im Standalone-Modus. Erstellen Sie nun die Tabellen MITARBEITER und URLAUB, indem Sie die entsprechenden SQL-Anweisungen des Kapitels im Textfeld des Database Managers eingeben und durch Drücken von "Execute" ausführen. Fügen Sie anschließend einige Mitarbeiter-Datensätze ein und manipulieren Sie diese wie im Kapitel ausgeführt. Wählen Sie mit SELECT-Anweisungen einige Datensätze aus. Um die Anzeige der Datenbankstruktur im linken Fenster zu aktualisieren, wählen Sie ***View*** → ***Refresh Tree*** aus.

Durch die Anweisungen

```
ALTER TABLE MITARBEITER ADD PRIMARY KEY ( PERSNR )
```

und

```
ALTER TABLE URLAUB ADD FOREIGN KEY ( PNR ) REFERENCES MITARBEITER ( PERSNR )
```

wird ein Primärschlüssel in der Tabelle MITARBEITER und ein Fremdschlüssel in der Tabelle URLAUB angelegt. Führen Sie diese aus und überprüfen Sie anschließend, welche Konsequenzen sich für die Datensätze aus den beteiligten Tabellen ergeben. Fügen Sie etwa mit

```
INSERT INTO URLAUB VALUES( 1006, '2006-01-31', '2006-02-14', 0)
```

einen Urlaubsdatensatz hinzu und versuchen Sie anschließend, den Mitarbeiterdatensatz mit der Personalnummer 1006 zu löschen.

24.2 HSQLDB: Server Mode und Standalone Mode

Die HSQLDB Datenbank kann im ***Server Mode*** oder im ***Standalone Mode*** gestartet werden. Im Standalone Mode (In-Process-Mode) wird die Datenbank Engine als Teil der Anwendung in der gleichen JVM betrieben. In diesem Modus ist es nicht möglich, sich von außerhalb an die Datenbank zu konnektieren. In Übung 24.1 wurde die Datenbank im Standalone Mode vom Database Manager gestartet, das Starten und Konnektieren eines weiteren Datenbank Managers schlägt daher fehl.

Wird eine HSQLDB Datenbank dagegen im Server Mode gestartet, können sich Anwendungsprogramme an diese konnektieren. Starten Sie die Datenbank im Server Mode mit dem Kommando

```
java -cp <hsqldb-Verzeichnis>/lib/hsqldb.jar  org.hsqldb.Server
```

sowie zwei Database Manager aus unterschiedlichen Konsolen (vgl. Übung 24.1).

Zum Konnektieren an die Datenbank passen Sie im ***Connect***-Fenster die folgenden Einträge an:

- Type: ***HSQL Database Engine Server***
- Driver: ***org.hsqldb.jdbcDriver*** (wird automatisch eingestellt)
- URL: ***jdbc:hsqldb:hsql://localhost/*** (wird automatisch eingestellt)

Verifizieren Sie, dass beide Database Manager an die gleiche Datenbank konnektiert sind, indem Sie mit einem eine Tabelle erstellen und die Views der Datenbankstruktur in beiden Managern aktualisieren.

Die Datenbank-Dateien befinden sich im Verzeichnis, in dem die JVM der Datenbank Enginge gestartet wurde und beginnen - wenn man keine weiteren Optionen mitgibt - mit dem Namen `test`. Weisen Sie nach, dass sich die in Übung 24.1 erstellte Datenbank auch im Server Mode starten lässt, indem man vor Eingabe des java-Kommandos in das Verzeichnis ***<Db-Daten-Verzeichnis>*** aus Übung 24.1 wechselt und das eben genannte Kommando zum Starten der Datenbank im Server Mode absetzt. (In Übung 24.1 wurden die Dateien explizit unter ***<Db-Daten-Verzeichnis>*** mit dem Namen `test` angelegt.)

24.3 SQL-Datendefinition und -Manipulation

Entwickeln Sie ein JDBC-Programm, das die Tabellen MITARBEITER und URLAUB erstellt und einige Datensätze hinzufügt. Hierzu sind die entsprechenden SQL-Anweisungen mit

```
st.execute( sqlString )
```

mit einem zuvor erzeugten Anweisungsobjekt `st` auszuführen.

In einer ersten Variante soll sich das Programm mit

```
DriverManager.getConnection( "jdbc:hsqldb:hsql://localhost", "sa", "" )
```

an eine zuvor im Server Mode gestartete Datenbank konnektieren, alternativ durch

```
DriverManager.getConnection( "jdbc:hsqldb:test", "sa", "" )
```

die Datenbank im Standalone Mode selbst starten.

Hinweis (nur für Standalone Mode):
Im Standalone Mode gibt es möglicherweise Ungereimtheiten im Zusammenhang mit der Schreibverzögerung der DB Engine. Dies zeigt sich darin, dass etwa die via SQL-Anweisung erstellten Tabellen beim erneuten Start der Datenbank nicht mehr vorhanden sind. In diesem Fall schafft eine der folgenden Lösungsalternativen abhilfe:

- zusätzliche Property im Datenbank-URL von `getConnection()`:
  "jdbc:hsqldb:test;shutdown=true"
- explizites Shutdown der Datenbank vor dem Schließen der Verbindung:
  `st.execute( "SHUTDOWN" );`
- Deaktivieren der Schreibverzögerung (vor dem Ausführen der eigentlichen Anweisungen):
  `st.execute( "SET WRITE_DELAY FALSE" );`

24.4 Textuelle Darstellung einer Tabelle

Formulieren Sie in einer ausführbaren Klasse die Methode

```
public static String table2String( Connection con, String tabName )
```

die eine textuelle Darstellung einer Tabelle bereit stellt, so dass die Tabelle MITARBEITER durch

```
Class.forName( "org.hsqldb.jdbcDriver" );
Connection con = DriverManager.getConnection( "jdbc:hsqldb:test", "sa", "" );
System.out.println( table2String( con, "MITARBEITER" ) );
```

in folgender Form ausgegeben wird (vgl. Beispiel des Kapitels)

```
MITARBEITER:
PERSNR       NACHNAME         VORNAME
- - - - - - - - - - - - - - - - - - - - - - - - -
1006         Meier            Rolf
1007         Lang             Ulla
1009         Keller           Ulrike
```

Für die Darstellung im Spaltenlayout verwendet man die statische Methode `format()` der Klasse `String` (vgl. [SUN05C]) mit konstanter Spaltenbreite.

Enthält ein Datensatz an einer Stelle den Wert NULL, so verhalten sich die `get()`-Methoden auf den ersten Blick ungewöhnlich, da sie einen Default-Wert (z.B. den Wert 0 bei `getInt()`) zurückgeben. Nach erfolgten `get()`-Aufruf kann jedoch mit der (parameterlosen!) Methode `wasNull()` kontrolliert werden, ob der Ausgabe ein NULL-Wert zugrunde lag oder nicht. Beachten Sie diesen Sachverhalt.

24.5 Metadaten der Datenbank

Zu einer bestehenden Datenbank-Verbindung können Informationen über die Datenbank mit

```
DatabaseMetaData dbm = con.getMetaData();
```

in Form einer Instanz von `DatabaseMetaData` verschafft werden. Neben Informationen über die unterstützten Leistungsmerkmale (z.B. ***outer-joins***, ***group-by***) erhält man daraus auch eine Beschreibung der aktuellen Datenbankobjekte (Tabellen, Indizes, Primärschlüssel etc.)

Geben Sie für das Beispiel im Kapitel den Datenbank-URL und alle Datenbanktabellen des Schemas "PUBLIC" aus, sowie für jede dieser Tabellen Informationen über eventuelle Primär- und Sekundärschlüssel. Eine Beschreibung der Methoden findet man in [SUN05C] unter ***DatabaseMetaData***.