Embedded SQL i Java

nikos dimitrakas
nikos@dsv.su.se

Connolly/Begg (3rd edition) Kapitel 21 + 28.8
(4th edition) Kapitel 29.7 + Appendix E
(5th edition) Kapitel 30.7.1, Appendix I
(6th edition) Kapitel 29.7.1, Appendix I

Vad är embedded SQL?

Värdspråk / Host Language

Valfritt programmingsspråk

Databas

En SQL-databas (relationsdatabas)

Embedded SQL

Hjälper värdspråket kommunicera med databasen.

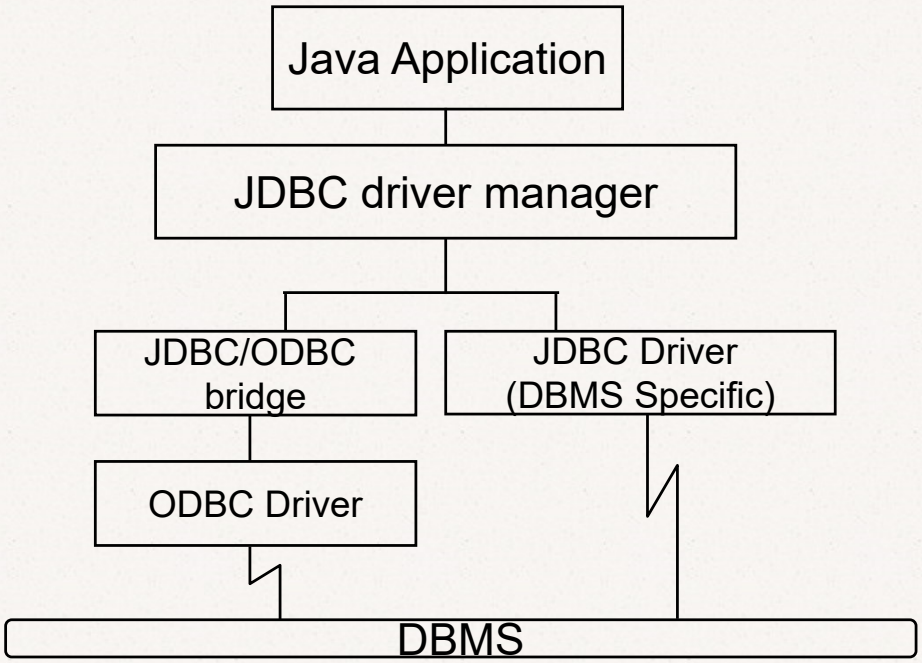
Varför embedded SQL?

- Gränssnitt mot databasen / Applikation
- Avancerad logik
- Jobba med flera databaser

Java embedded SQL

- Javaprogram
- JDBC-driver
- (JDBC-ODBC bridge + ODBC-driver) (borta i Java 8)
- Databas

Java embedded SQL Arkitektur



JDBC-programsekvens

1. Importerta paket
2. Öppna en Connection till databasen
3. Skapa en Statement (eller PreparedStatement)
4. Exekvera en SQL-sats och ta emot resultatet (ett ResultSet) om resultat finns
5. Jobba med resultatet (om det finns)
6. Stäng ResultSet och Statement
7. Stäng Connection

Connection kan återanvändas genom att göra punkterna 3-6 flera gånger.

Connection URL

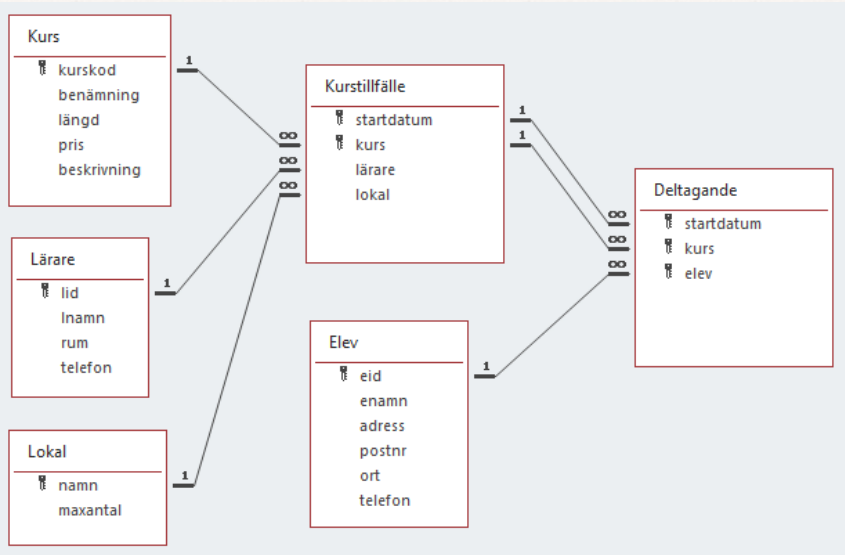
En URL ("databasadress") består av tre delar:

jdbc:databastyp:databasnamn

- **jdbc:mysql:///labb**
  - **Lokal mysql-server och en databas/schema som heter "labb"**
- **jdbc:mysql://*serveradress*/labb**
  - **serveradress kan vara ip-adress eller namn (och även port)**
- **jdbc:ucanaccess://filnamn**
  - **filnamn är hela sökvägen, t ex c:/db/labb.accdb**
- **jdbc:odbc:labb ("labb" är ett ODBC-alias)**
  - **ODBC-alias kan skapas i ODBC Administrator (Windows)**
  - **Lämpligt för databaser som inte har en Java-driver**
  - **Funkar inte fr o m Java 8**
  - **Primärt i Windows**

Databasen

Följande databas används i exempelkoden (och för labben)



1. Importera Paket

```
// Import packages
import java.sql.*; //JDBC packages

// Other imports
import java.util.Scanner;
…
```

java.sql package specification:
http://docs.oracle.com/javase/8/docs/api/

2. Öppna en Connection till databasen

```
// DB access variables
String URL = "jdbc:ucanaccess://SQLDatabas.accdb";

// Create a connection to the database
Connection con = DriverManager.getConnection(URL);
```

Det finns även en getConnection som tar emot användarnamn och lösenord som parametrar.

3. Skapa en Statement (eller PreparedStatement)

```
// Create a statement associated to the connection con.
// The new statement is placed in the variable stmt.
Statement stmt = con.createStatement();
```


4. Exekvera en SQL-sats och ta emot resultatet

```
// Set the SQL statement into the variable query
String query = "SELECT ort, COUNT(*) AS antal
                FROM Elev
                GROUP BY ort";

// Execute the SQL statement that is stored in the variable
// query and store the result in the variable rs.
ResultSet rs = stmt.executeQuery(query);
```

5. Jobba med resultatet

```
// Loop through the ResultSet and print the results.
// The method next() returns false when there are no more rows.
while (rs.next()) {
   System.out.print("Ort: " + rs.getString("ort"));
   System.out.println(" Antal elever: " + rs.getInt("antal"));
}
```


6. Stäng ResultSet och Statement

```
// Close the variable stmt and release all resources
// bound to it.
// Any ResultSet associated to the Statement will be
// automatically closed too.
stmt.close();


// Since Java 7, you could use try with resources
// instead of manually closing statements.
```

3. Skapa en Statement (eller PreparedStatement)

```
// Set the SQL statement into the variable query
String query = "SELECT enamn, telefon
                FROM Elev
                WHERE ort = ?
                AND eid IN (SELECT elev
                            FROM Deltagande
                            WHERE kurs = ?)";

// Create a PreparedStatement associated to
// the connection and the query.
// The new statement is placed in the variable pstmt.
PreparedStatement pstmt = con.prepareStatement(query);
```


4. Exekvera en SQL-sats och ta emot resultatet

```
String ortparam = "Kista";
String kursparam = "Java1";

 // Provide the value for the first ? in the SQL statement.
pstmt.setString(1, ortparam);
// Provide the value for the second ? in the SQL statement.
pstmt.setString(2, kursparam);

// Execute the SQL statement that is prepared in the
// variable pstmt and store the result in the variable rs.
ResultSet rs = pstmt.executeQuery();
```

5. Jobba med resultatet

```
// Loop through the ResultSet and print the results.
// The method next() returns false when there are no more rows.
while (rs.next()) {
   System.out.println(rs.getString("enamn") + " " +
   rs.getString("telefon"));
}
```


6. Stäng ResultSet och Statement

```
// Close the variable pstmt and release all resources
// bound to it.
// Any ResultSet associated to the Statement will be
// automatically closed too.
pstmt.close();


// Since Java 7, you could use try with resources
// instead of manually closing statements.
```

7. Stäng Connection

```
// Close the connection
con.close();


// Since Java 7, you could use try with resources
// instead of manually closing connections.
```


Datatyper

Datatyper i databasen måste kunna mappas till datatyper/klasser i Java:

- SQL integer, etc → Java int (eller short/long)
- SQL number, real, etc → Java float/double/BigDecimal
- SQL varchar, char, string, etc → Java String
- SQL date, time, timestamp, etc → Java java.sql.Date, java.sql.Time, java.sql.Timestamp
- SQL bit, boolean → Java boolean

Läs mer i API för java.sql.ResultSet eller på
http://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/index.html

SQL-satser utan resultat

Statement och PreparedStatement har även metoder som executeUpdate(String query) och executeUpdate() som är lämpliga när man inte förväntar sig en ResultSet, dvs för andra SQL-satser än SELECT.

SQL Injection attacks

- Konkatenering av användarinput i SQL-satser
  - String sql = "UPDATE Person SET epost = '"+epost+"' WHERE pnr = '"+pnr+"'"
  - Vad händer om pnr är

    ' OR '1'='1
  - String sql = "SELECT * FROM Person WHERE epost = '"+mail+"' AND lösenord = '"+password+"'"
  - Vad händer om password är

    ' OR epost='rektor@su.se
- Parametrisering skyddar
  - String sql = "UPDATE Person SET epost = ? WHERE pnr = ?"
  - String sql = "SELECT * FROM Person WHERE epost = ? AND lösenord = ?"
  - Parametrisera och exekvera med PreparedStatement