Aplicații integrate pentru întreprinderi

Laborator 3

19.10.2010

Manipularea informațiilor dintr-o bază de date MySQL folosind Java

Scopul laboratorului îl reprezintă utilizarea unor comenzi MySQL (precum şi rezultatul lor) prin intermediul limbajului de programare Java, folosind un API care respectă protocolul Java Database Connectivity (JDBC).

1. Ce este un *„driver”* pentru un sistem de gestiune al bazei de date ?

Un *„driver”* pentru un sistemul de gestiune al bazei de date este o librărie prin care sunt transformate apelurile JDBC (din limbajul de programare Java) într-un format suportat de protocolul de rețea folosit de sistemul de gestiune al bazei de date, permițând programatorilor să acceseze datele din medii eterogene.

În cadrul laboratorului (în vederea rezolvării temei de casă), pentru conectarea la baza de date MySQL, vom folosi un produs denumit **Connector/J**, driver nativ pentru Java dezvoltat de compania Oracle distribuit în mod gratuit utilizatorilor.

Paradigma pe care o vom folosi implică două niveluri şi este descrisă în Figura 1:

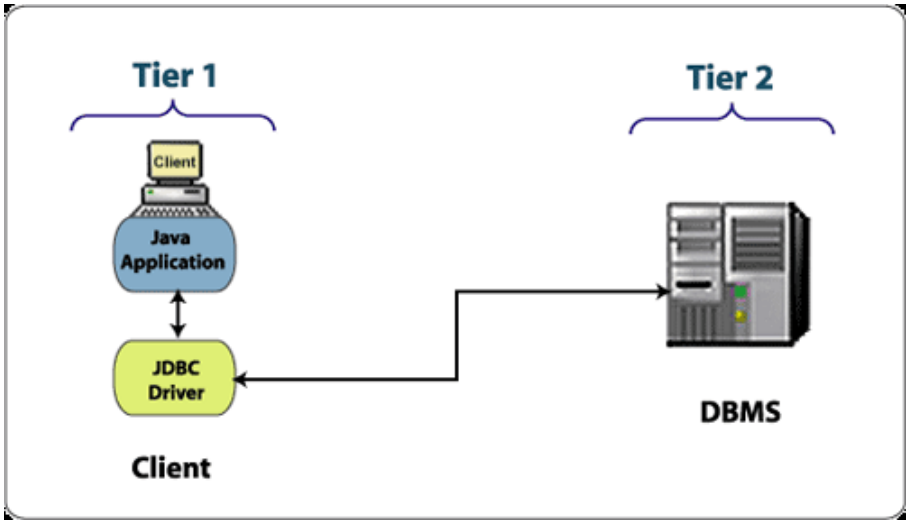

Figura 1 – Conectarea dintr-o aplicație Java la sistemul de gestiune al bazei de date prin intermediul unui driver JDBC

2. Configurare Connector/J

Tot ce trebuie făcut pentru utilizarea driver-ului de conectare din limbajul de programare Java la sistemul de gestiune al bazei de date (oricare ar fi: MySQL) este să descărcați arhiva care conține Connector/J[1] de pe pagina oficială (http://www.mysql.com/downloads/connector/j/), să o dezarhivați şi să adăugați fişierul .jar din rădăcină la classpath în momentul în care compilați aplicația.

În linie de comandă, acest lucru poate fi realizat astfel:

- compilare:

```
>javac -classpath .;mysql-connector-java-5.1.13-bin.jar filename.java
```

- rulare:

```
>java -classpath .;mysql-connector-java-5.1.13-bin.jar filename
```

Mai uşor, puteți folosi medii integrate de dezvoltare a aplicațiilor, cum ar fi

- *Eclipse* (a fost testată versiunea 3.5.2)
    - bibliotecile conținute în arhiva .jar trebuie adăugate la calea proiectului prin comanda *„Add external libraries”* (click dreapta pe numele proiectului -> Build Path) – Figura 2
    - dacă librăria externă a fost adăugată în mod corect, numele librăriei trebuie să apară în meniul din stânga corespunzător proiectului, secțiunea *„Referenced libraries”* – Figura 3
- *NetBeans* (a fost testată versiunea 6.9)
    - în meniul din stânga corespunzător proiectului, alegeți *Libraries*, apoi click dreapta şi selectați opțiunea *Add JAR/Folder* – Figura 4
    - va apărea o fereastră de dialog în care aveți grijă să selectați referința drept cale absolută (Refernce As: -> Absolute Path), în caz contrar, includeți arhiva în sistemul de fişiere al proiectului – Figura 5

[1] Ultima versiune disponibilă pe pagina oficială este 5.1.13.

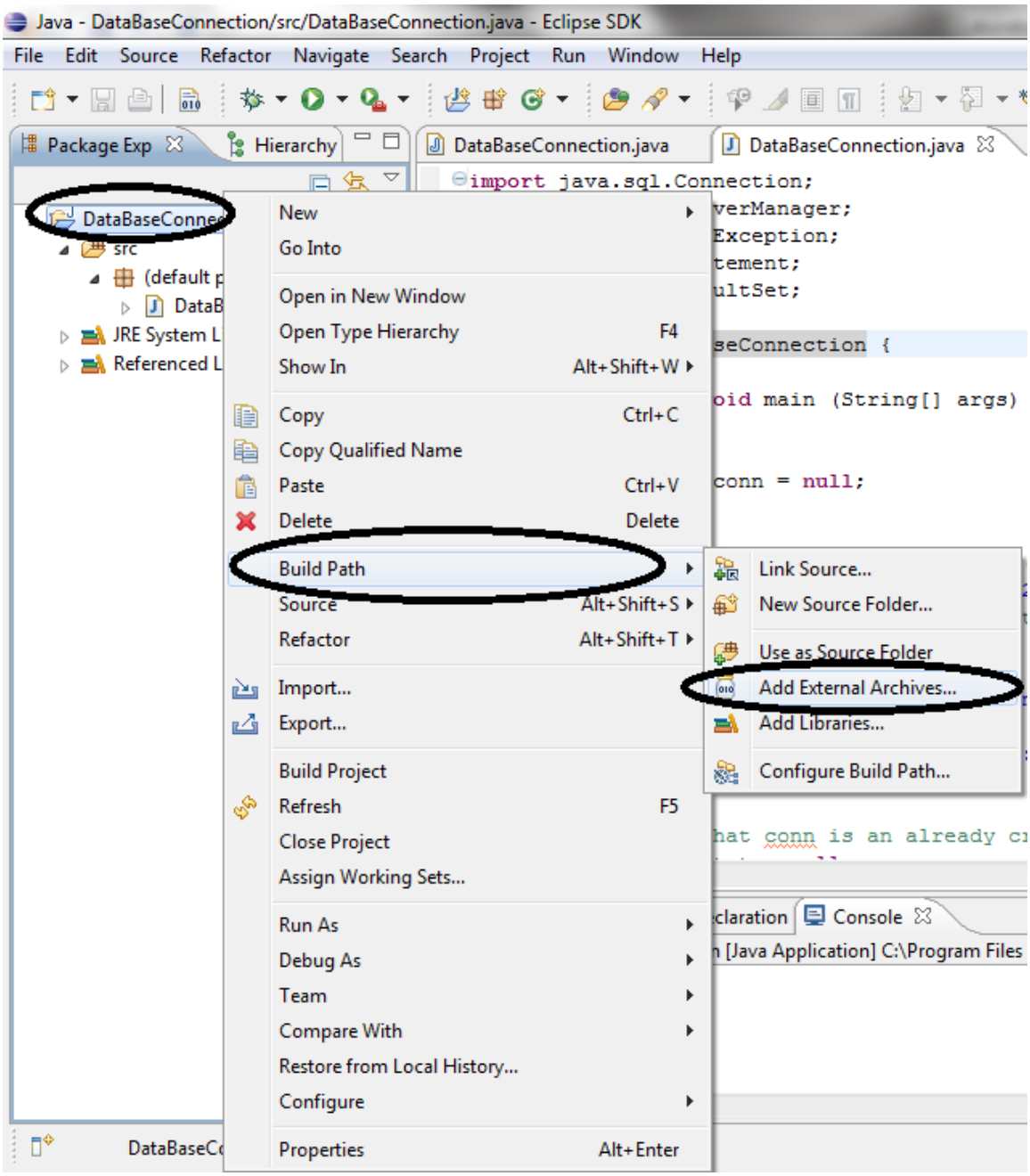

Figura 2 – Adăugarea unei biblioteci externe pentru un proiect în *Eclipse 3.5.2*

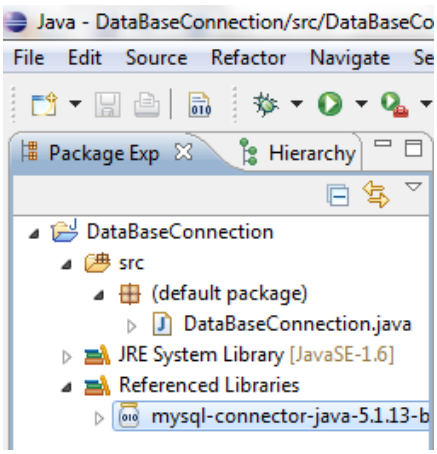

Figura 3 – Adăugarea corectă a bibliotecii externe este marcată prin adăugarea ei în meniul din stânga aferent proiectului, secțiunea *Referenced Libraries*

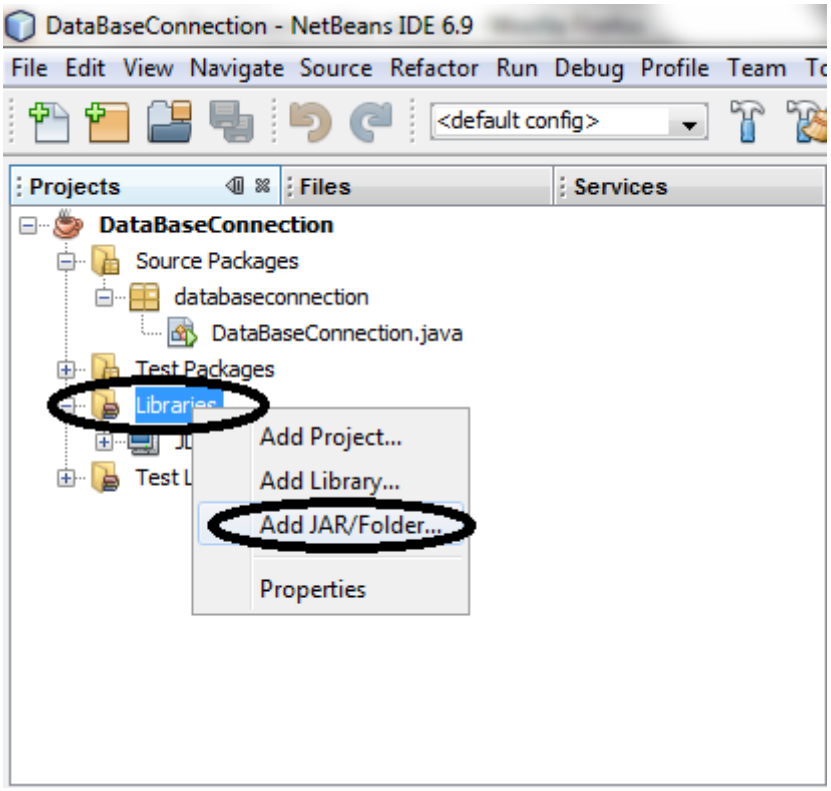

Figura 4 – Adăugarea unei biblioteci externe pentru un proiect în *NetBeans 6.9*

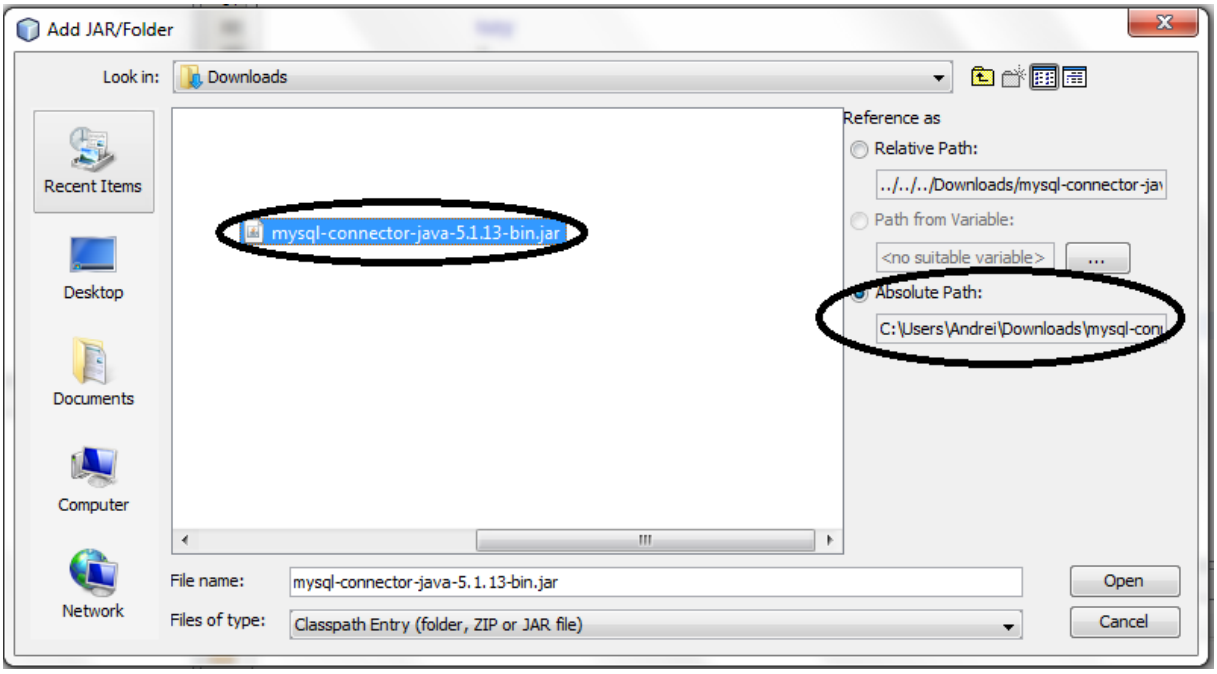

Figura 5 – Referirea căii absolute către librăria externă în condițiile în care nu este inclusă în sistemul de fişiere al proiectului

3. Arhitectura JDBC

Arhitectura protocolului JDBC, descrisă în Figura 6, este structurată pe două niveluri:

- un API JDBC responsabil de comunicația dintre aplicația Java şi modulul de gestiune al driver-ului;
- un API JDBC Driver care este responsabil de comunicația dintre modulul de gestiune al driver-ului şi baza de date; un astfel de nivel este independent atât în raport cu baza de date la care se conectează precum şi în raport cu limbajul de programare din care este accesat;

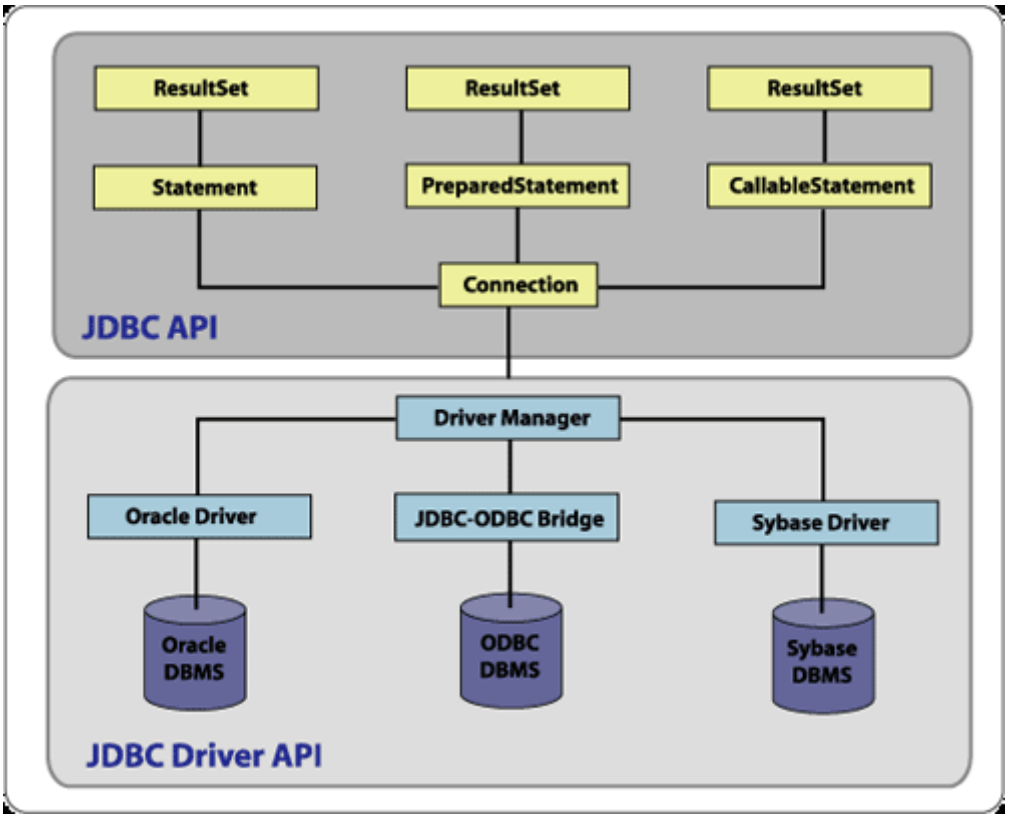

Figura 6 – Arhitectura protocolului JDBC

4. Conectarea la sistemul de gestiune al bazei de date

Conectarea unei aplicații Java prin intermediul protocolului JDBC la sistemul de gestiune al bazei de date se poate realiza prin două clase:

- `DriverManager` – presupune încărcarea unui anumit driver prin indicarea unei resurse de localizare de tip URL;
- `DataSource` – metodă mai transparentă de acces la informații, având proprietăți specificate încât să corespundă unor surse de date particulare;

În cadrul laboratorului vom folosi metoda de conectare la baza de date folosind clasa `DriverManager` întrucât este mai intuitivă. Atunci când un client indică un URL pentru a se conecta la o bază de date, clasa `DriverManager` apelează la interfața `Driver` pentru a identifica driver-ul necesar pentru interacțiunea cu sistemul de gestiune al bazei de date.

De obicei, formatul URL-ului respectă următoarea structură:

```
protocol:subprotocol:<nume_bazadedate>[lista_de_proprietati]
```

Un exemplu de URL valid (care respectă formatul de mai sus) este:

```
jdbc:mysql://localhost/mysql?user=root&password=student
```

Metoda `getConnection`, disponibilă în clasa `DeviceManager` oferă un obiect conexiune (`Connection`) la baza de date, care poate fi folosit ulterior pentru diferite interogări:

```
String URL = jdbc:mysql://localhost/mysql;
Connection connection = DriverManager.getConnection(URL);
```

În condițiile în care se citesc dintr-o interfață grafică cu utilizatorul informații de tip utilizator şi parolă, stocate, spre exemplu în variabilele `usr` şi `pwd`, conectarea se poate face şi sub următoarea formă:

```
Connection connection = DriverManager.getConnection(URL,usr,pwd);
```

sau

```
Connection connection =
         DriverManager.getConnection(URL+"?user="+usr+"&password="pwd);
```

Atunci când driverele gestionate de interfața `Driver` recunosc URL-ul dat ca parametru metodei `getConnection`, se stabileşte o legătură cu sistemul de gestiune pentru baza de date, întorcându-se o conexiune deschisă ce poate fi utilizată pentru formularea de instrucțiuni JDBC care să transmită interogări către baza de date.

5. Interogarea bazei de date conform specificației JDBC

Tipurile de interogări pot fi, conform specificației JDBC, de trei feluri:

- `Statement`
- `PreparedStatement`
- `CallableStatement`

Un obiect de tip interogare (`Statement`[2]) se obține prin metoda `createStatement` aplicabilă unui obiect de tip `Connection`:

```
Statement statement = connection.createStatement();
```

În continuare, obiectul de tip interogare poate fi utilizat pentru realizarea unei operații cu baza de date şi obținerea unui set de date rezultat în urma executării instrucțiunii:

```
String query = „SELECT * from users";
ResultSet rs = statement.executeQuery (query);
```

[2] Aşa cum vom vedea în continuare, obiectele de tip `PreparedStatement` sunt utilizate atunci când nu sunt cunoscute toate datele interogării, acestea devenind disponibile de abia la run-time.

Interfața `ResultSet` pune la dispoziția utilizatorului o serie de metode pentru lucrul cu informațiile obținute în urma interogării bazei de date. Obiectele având tipul `ResultSet` au anumite caracteristici care pot fi modificate între care tipul, gestiunea concurenței şi posibilitatea de deținere a cursorului. Astfel de opțiuni pot fi precizate de utilizator în momentul creării unui obiect de tip interogare (`Statement`).

Cu privire la modalitatea în care poate fi manipulat cursorul (aspect ce ține de senzitivitatea cursorului), există următoarele constante:

- `TYPE_FORWARD_ONLY` – cursorul se poate muta doar înainte, ne-existând posibilitatea parcurgerii în ambele sensuri a setului de date obținut ca urmare a interogării;
- `TYPE_SCROLL_INSENSITIVE` – cursorul se poate muta înainte şi înapoi, poziționându-se în diferite locații relative de poziția curentă sau absolute, dar nu este afectat de modificările realizate de alții;
- `TYPE_SCROLL_SENSITIVE` – cursorul se poate muta înainte şi înapoi, poziționându-se în diferite locații relative de poziția curentă sau absolute, şi este afectat de modificările realizate de alții;

Tipul de concurență indică operațiile pe care utilizatorul are permisiunea spre a le realiza:

- `CONCUR_READ_ONLY` – utilizatorul are doar dreptul de a consulta informațiile, fără a le modifica;
- `CONCUR_UPDATABLE` – utilizatorul poate citi şi poate scrie informațiile reținute în setul rezultat;

Un exemplu de creare a unui obiect de tip interogare realizat pentru obținerea unui set de date în care cursorul poate fi mutat în ambele direcții, dar nu poate fi modificat este redat mai jos:

```
Statement statement =
       connection.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                  ResultSet.CONCUR_READ_ONLY);
```

Un obiect de tip `ResultSet` conține mai multe (sau nici un) rând(uri), în funcție de specificitatea interogării, având asociat un cursor care indică la orice moment rândul curent[3].

Unele dintre metodele care pot fi utilizate pentru a realiza poziționări ale cursorului în cadrul setului de date sunt:

metoda	descriere
next()	mută cursorul pe înregistrarea următoare
previous()	mută cursorul pe înregistrarea precedentă
first()	mută cursorul pe prima înregistrare
last()	mută cursorul pe ultima înregistrare
beforeFirst()	mută cursorul înainte de prima înregistrare
afterLast()	mută cursorul după prima înregistrare
relative (int n)	mută cursorul la n poziții distanță față de poziția curentă
absolute (int n)	mută cursorul la poziția n (absolută) din set

[3] Inițial, cursorul se găseşte deasupra primului rând.

Obținerea informațiilor (valorilor asociate coloanelor) este realizată prin aşa-numitele metode *getter* (getString[4], getInt, getByte, getBoolean, getBlob, getDate) care pot primi ca parametru fie numele coloanei[5] fie indexul[6] ei în cadrul tabelei din baza de date.

O rutină de parcurgere a tuturor înregistrărilor dintr-o bază de date poate fi cea descrisă mai jos:

```
ResultSet source = statement.executeQuery
                    ("SELECT cota, titlu, autor FROM carti");
while (source.next())
// move the cursour to next record and test if it exists
{
     float cota = source.getFloat("cota");
     String titlu = source.getString(2);
     String autor = source.getString("autor");
}
```

Procesul de actualizare a informațiilor reținute într-o bază de date e realizat în două etape:

- modificarea valorilor ce se doresc actualizate, la nivel de coloană, pe rândul unde se găseşte cursorul, prin intermediul metodelor `update...()`; în acest moment, nici o modificare nu este marcată la nivelul tabelei;
- actualizarea rândului curent în care au fost marcate spre modificare valorile coloanelor prin intermediul metodei `updateRow()`;

Un exemplu de actualizare a informațiilor în baza de date este redat mai jos:

```
Statement statement = connection.createStatement
                          (ResultSet.TYPE_SCROLL_SENSITIVE,
                           ResultSet.CONCUR_UPDATABLE);
ResultSet source = statement.executeQuery
                    ("SELECT cota, titlu, autor, exemplare FROM carti");
while (source.next())
// move the cursour to next record and test if it exists
{
     source.updateFloat(exemplare,1);
     updateRow();
}
```

6. Utilizarea interogărilor parametrizabile

Atunci când nu toate datele interogării sunt cunoscute la momentul în care progamul este compilat, există posibilitatea ca interogarea să fie generică, urmând a fi completată cu informații (provenite dintr-un fişier sau introduse chiar de către utilizator) atunci când ele sunt disponibile, şi anume la rulare, înainte de execuția interogării asupra bazei de date.

În acest sens, sunt folosite obiecte de tip `PreparedStatement`, derivate din clasa `Statement`, care primesc la momentul construcției şi interogarea corespunzătoare, în care informațiile necunoscute sunt înlocuite prin caracterul `?`:

---

[4] Metoda poate fi folosită pentru preluarea oricărui tip de informație din baza de date, mai puțin tipul SQL 3.

[5] Metoda nu ține cont de capitalizarea şirului de caractere care este oferit drept parametru.

[6] Această metodă este mai eficientă. Numerotarea coloanelor începe de la 1.

```
String query = UPDATE cărţi SET exemplare = ? WHERE autor= ?;
PreparedStatement statement = connection.prepareStatement (query);
```

Înainte de a executa o astfel de interogare, trebuie specificate valorile care corespund elementelor lipsă, lucru care se face prin metode de tip *setter*:

```
statement.setInt (1, Integer.parseInt(jscrollbar.getValue()));
statement.setString (2, jtextarea.getTextArea().getText());
```

Execuția interogării odată ce toate datele sunt cunoscute se face prin comanda executeUpdate[7]:

```
statement.executeUpdate();
connection.commit()[8];
```

[7] Dacă interogarea întoarce un singur set de date, se foloseşte instrucțiunea `executeQuery`, iar pentru situațiile în care este posibil să se obțină mai multe seturi de date să se folosească doar instrucțiunea `execute`.

[8] Este important ca la începutul tranzacțiilor să se apeleze `connection.setAutoComit(false)` pentru a nu se produce modificări în baza de date până când acest lucru nu este specificat explicit