# LUCRUL CU BAZE DE DATE ÎN JAVA

***Lect.univ.dr.ing. IOAN-GHEORGHE RAȚIU***
***Lect.univ. NICOLETA DAVID***

*Universitatea „George Barițiu“, Braşov*

## Rezumat

*O bază de date reprezintă o modalitate de stocare a unor informații (date) pe un suport extern, cu posibilitatea regăsirii acestora. O bază de date este memorată într-unul sau mai multe fişiere care sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date (SGBD). Cel mai răspândit model de baze de date este cel relațional, în care datele sunt memorate în tabele. Pe lângă tabele, o bază de date relațională mai poate conține: indecşi, proceduri stocate, trigger-e, utilizatori şi grupuri de utilizatori, tipuri de date, mecanisme de securitate şi de gestiune a tranzacțiilor etc. Alte modele de baze de date sunt modelul ierarhic, modelul orientat-obiect şi modelul XML.*

### 1. Generalități despre baze de date

Aplicațiile care folosesc baze de date sunt, în general, aplicații complexe folosite pentru gestionarea unor informații de dimensiuni mai mari într-o manieră sigură şi eficientă.

O *bază de date* reprezintă o modalitate de stocare a unor informații (date) pe un suport extern, cu posibilitatea regăsirii acestora. Uzual, o bază de date este memorată într-unul sau mai multe fişiere care sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date (SGBD). Modelul clasic de bază de date este cel relațional, în care datele sunt memorate în tabele. Pe lângă tabele, o bază de date mai poate conține: indecşi, proceduri stocate, trigger-e, utilizatori şi grupuri de utilizatori, tipuri de date, mecanisme de securitate şi de gestiune a tranzacțiilor etc.

Alte modele de baze de date sunt: modelul ierarhic, modelul orientat-obiect şi modelul XML.

XML (e**X**tensible **M**arkup **L**anguage) este un meta-limbaj de marcare recomandat de către Consorțiul Web pentru crearea de alte limbaje de marcare (marcaj sau tag – acțiunea de interpretare explicită a unei porțiuni de text), cum ar fi XHTML, RDF, RSS, MathML, SVG, OWL etc. Aceste limbaje formează familia de limbaje XML. Meta-limbajul XML este o simplificare a limbajului SGML (din care provine şi HTML) şi a fost proiectat în scopul transferului de date între aplicații pe Internet. XML este acum şi un model de stocare a datelor nestructurate şi semistructurate în cadrul Bazelor de date native XML.

Producătorii cei mai importanți de baze de date sunt: Oracle, Sybase, IBM, Informix, Microsoft etc.

Crearea unei baze de date se face cu aplicații specializate oferite de producătorul tipului respectiv de bază de date.

Accesul la o bază de date se face prin intermediul unui driver specific tipului de bază de date. Acesta este responsabil cu accesul efectiv la datele stocate, fiind legătura între aplicație şi baza de date.

**2. Driverul JDBC**

*JDBC (Java Database Connectivity)* este o interfață standard SQL de acces la baze de date. JDBC este constituită dintr-un set de clase şi interfețe scrise în Java, furnizând mecanisme standard pentru proiectanții aplicațiilor de baze de date.

Pachetul care oferă suport pentru lucrul cu baze de date este **java.sql**.

Folosind JDBC este uşor să transmitem secvențe SQL către baze de date relaționale. Cu alte cuvinte, nu este necesar să scriem un program pentru a accesa o bază de date Oracle, alt program pentru a accesa o bază de date Sybase şi aşa mai departe. Este de ajuns să scriem un singur program folosind API-ul JDBC şi acesta va fi capabil să trimită secvențe SQL bazei de date dorite. Bineînțeles, scriind codul sursă în Java, ne este asigurată portabilitatea programului. Acestea sunt două motive puternice care fac combinația Java – JDBC demnă de luat în seamă.

Fiind robust, sigur, uşor de folosit şi uşor de înțeles, Java este un excelent limbaj pentru a dezvolta aplicații de baze de date. Ceea ce-i lipseşte este modalitatea prin care aplicațiile Java pot comunica cu bazele de date. JDBC-ul însă oferă acest mecanism.

În linii mari, JDBC face trei lucruri:

– stabileşte o conexiune cu o bază de date;
– trimite secvențe SQL;
– prelucrează rezultatele.

**3. Conectarea la o bază de date**

Procesul de conectare la o bază de date implică două operații:

– încărcarea în memorie a unui driver corespunzător;
– realizarea unei conexiuni propriu-zise.

O *conexiune (sesiune)* la o bază de date reprezintă un context prin care sunt trimise secvențe SQL şi primite rezultatele. Într-o aplicație pot exista mai multe conexiuni simultan la baze de date diferite sau la aceeaşi bază.

Clasele şi interfețele pentru realizarea unei conexiuni sunt:

– clasa **DriverManager**, care se ocupă cu înregistrarea driverelor ce vor

fi folosite în aplicație;

– interfața **Driver**, pe care trebuie să o implementeze orice clasă ce descrie un driver;

– clasa **DriverPropertyInfo**;

– interfața **Connection**, descrie obiectele ce modelează o conexiune propriu-zisă cu baza de date.

*Încărcarea în memorie a unui driver*

Primul lucru pe care trebuie să-l facă o aplicație în procesul de conectare la o bază de date este să încarce în memorie clasa care implementează driver-ul necesar comunicării cu respectiva bază de date.

Acest lucru poate fi realizat prin mai multe modalități:

– DriverManager.registerDriver(newsun.jdbc.odbc.JdbcOdbcDriver());
– Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
– System.setProperty("jdbc.drivers","sun.jdbc.odbc.JdbcOdbcDriver");
– java-Djdbc.drivers=sun.jdbc.odbc.JdbcOdbcDrive.

*Specificarea unei baze de date*

Îndată ce un driver JDBC a fost încărcat în memorie cu DriverManager, acesta poate fi folosit la stabilirea unei conexiuni cu o bază de date.

Având în vedere faptul că pot exista mai multe drivere înregistrate în memorie, trebuie să avem posibilitea de a specifica pe lângă identificatorul bazei de date şi driverul ce trebuie folosit.

Aceasta se realizează prin intermediul unei adrese specifice, numită JDBC URL, ce are următorul format:

**jdbc:sub-protocol:identificator_baza_de_date**

Câmpul *sub-protocol* denumeşte tipul de driver care trebuie folosit pentru realizarea conexiunii şi poate fi *odbc, oracle, sybase, db2* etc.

*Identificatorul bazei de date* este un indicator specific fiecărui driver care specifică baza de date cu care aplicația doreşte să interacţioneze.

În funcție de tipul driver-ului acest identificator poate include numele unei maşini-gazdă, un număr de port, numele unui fişier sau al unui director etc.:

– jdbc:odbc:testdb;
– jdbc:oracle:thin@persistentjava.com:1521:testdb;
– jdbc:sybase:testdb;
– jdbc:db2:testdb.

La primirea unui JDBC URL, DriverManager-ul va parcurge lista driver-elor înregistrate în memorie, până când unul dintre ele va recunoaşte URL-ul respectiv.

Dacă nu există niciunul potrivit, atunci va fi lansată o excepție de tipul SQLException, cu mesajul no suitable driver.

*Realizarea unei conexiuni*

Metoda folosită pentru realizarea unei conexiuni este getConnection din clasa DriverManager şi poate avea mai multe forme:

– Connection c = DriverManager.getConnection(url);

– Connection c = DriverManager.getConnection(url, username, password);

– Connection c = DriverManager.getConnection(url, dbproperies).

O conexiune va fi folosită pentru:

– crearea de secvențe SQL ce vor fi folosite pentru interogarea sau actualizarea bazei de date;

– aflarea unor informații legate de baza de date (meta-date).

Clasa Connection asigură suport pentru controlul tranzacțiilor din memorie către baza de date prin metodele commit, rollback, setAutoCommit.

**4. Efectuarea de secvențe SQL**

O dată făcută conectarea cu DriverManager.getConection(), se poate folosi obiectul Connection rezultat pentru a se crea un obiect de tip Statements, cu ajutorul căruia se pot trimite secvențe SQL către baza de date. Cele mai uzuale comenzi SQL sunt cele folosite pentru:

– interogarea bazei de date (SELECT);

– actualizarea bazei de date (INSERT, UPDATE, DELETE):

```
Connection c = DriverManager.getConnection(url);
Statement s = c.createStatement();
ResultSet r = s.executeQuery("SELECT * FROM un_tabel ORDER
BY o_coloana");
s.executeUpdate("DELETE * FROM un_tabel").
```

Metoda executeQuery trimite interogări SQL către baza de date şi primeşte răspuns într-un obiect de tip ResultSet.

**5. Obținerea şi prelucrarea rezultatelor**

*Interfața ResultSet*:

```
String query = "SELECT cod, nume FROM localități ORDER BY
nume";
ResultSet r = s.executeQuery( query );
while (r.next()) {
System.out.println (
r.getString ("cod") + "," +
```

r.getString ("nume")).

*Interfața ResultSetMetaData*:

```
ResultSet r = s.executeQuery("SELECT*FROM localitati");
ResultSetMetaData rsmd = r.getMetaData();
System.out.println("Coloane: " + rsmd.getColumnCount()).
```

**Concluzii**

Accesul la o bază de date se face prin intermediul unui driver specific tipului respectiv de bază de date. Acesta este responsabil cu accesul efectiv la datele stocate, fiind legătura între aplicație şi baza de date. Procesul de conectare la o bază de date implică două operații:

- încărcarea în memorie a unui driver corespunzător;
- realizarea unei conexiuni propriu-zise.

Accesul la baza de date se face cu ajutorul *JDBC (Java Database Connectivity),* o interfață standard SQL, constituită dintr-un set de clase şi interfețe scrise în Java, care furnizează mecanisme standard pentru proiectanții aplicațiilor de baze de date.

După conectarea cu DriverManager.getConection(), se poate folosi obiectul Connection rezultat pentru a se crea un obiect de tip Statements, cu ajutorul căruia se pot trimite secvențe SQL către baza de date.

## BIBLIOGRAFIE


1. Tănasă, Ştefan, Olaru, Cristian, Andrei, Ştefan*, Java de la 0 la Expert*, Iaşi, Editura Polirom, 2003
2. Velicanu, Manole, Lungu, Ion ş.a., *Sisteme de baze de date – teorie şi practică*, Bucureşti, Editura Petrion, 2003