Curs 9 PPOO

Prof. univ. dr. Cristian CIUREA

Departamentul de Informatică și Cibernetică Economică

cristian.ciurea@ie.ase.ro

Java fundamentals

- Lucrul cu baze de date în Java
- Arhitectura JDBC
- Operații DDL/DML

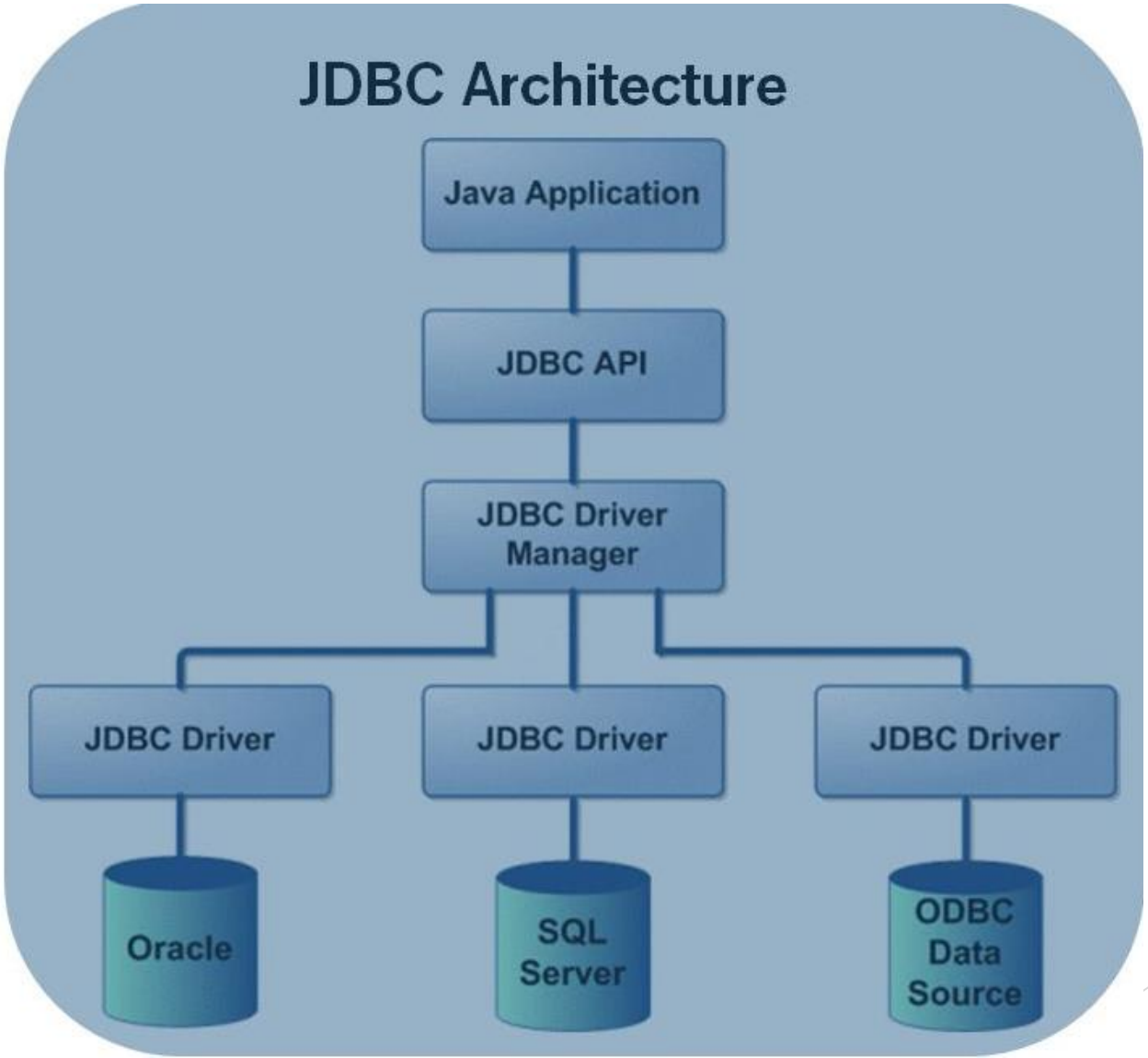
Lucrul cu baze de date în Java

- Instalarea ultimei versiuni de Java SE SDK
- Instalarea sistemului de gestiune a bazelor de date (DBMS) în cazul în care este necesar:
  - **Java DB**
    - http://www.java2s.com/Code/Jar/d/Downloadderbyclient10910jar.htm
  - **MySQL**
    - http://www.java2s.com/Code/Jar/m/Downloadmysqlconnectorjar.htm
  - **SQLite**
    - http://www.java2s.com/Code/Jar/s/Downloadsqlitejdbc372jar.htm

Lucrul cu baze de date în Java

- Instalarea și configurarea unui driver JDBC de la furnizorii bazelor de date utilizate:
  - org.apache.derby.jdbc.ClientDriver
  - com.mysql.jdbc.Driver
  - org.sqlite.JDBC
  - oracle.jdbc.driver.OracleDriver

Arhitectura JDBC



Lucrul cu baze de date în Java

- O aplicație JDBC utilizează unul sau mai multe drivere din pachetul **java.sql.*** care sunt utilizate de către clasa **DriverManager**.
- Driverele sunt specifice bazelor de date, deci pentru fiecare tip de bază de date se utilizează un driver special.
- În aceeași aplicație putem lucra cu baze de date diferite, deci implicit și cu mai multe drivere.

Lucrul cu baze de date în Java

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

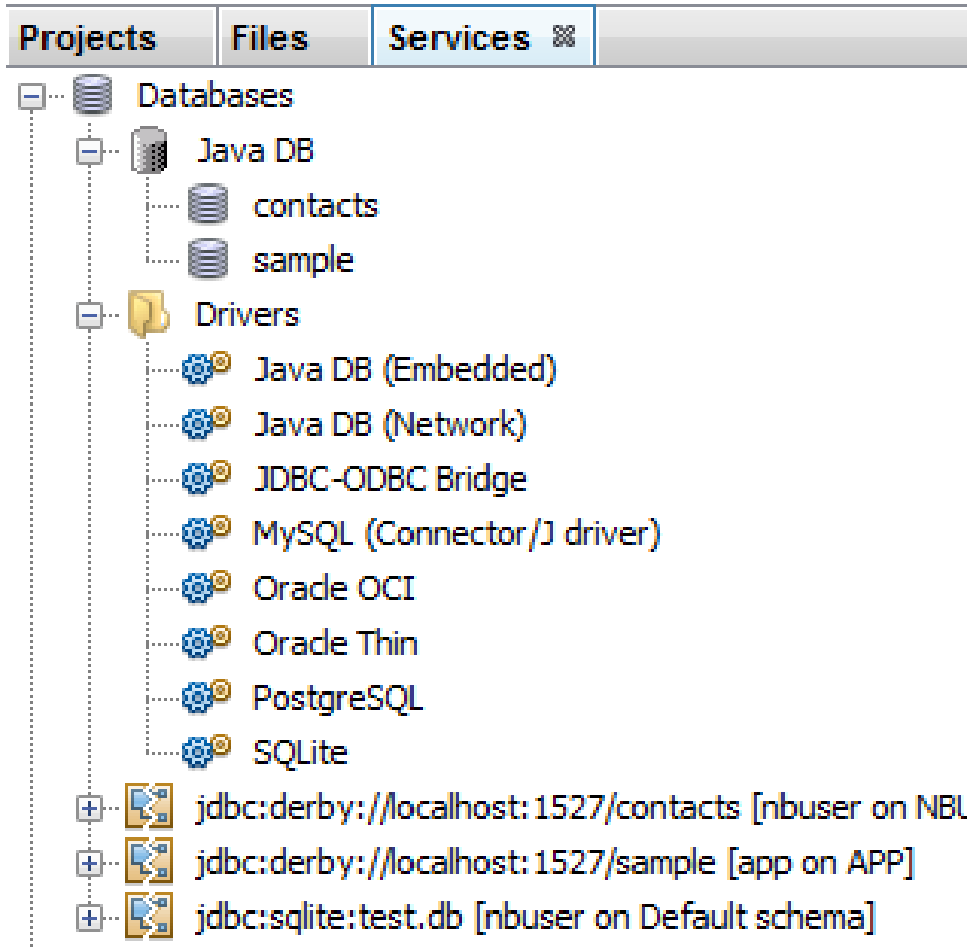
- încărcarea în memorie a unui driver corespunzător (**DriverManager**);
- realizarea unei conexiuni propriu-zise:
  - **DriverManager.getConnection(url);**
  - **DriverManager.getConnection(url, username, password);**
  - **DriverManager.getConnection(url, dbproperies);**

Lucrul cu baze de date în Java

- Baza de date **Java DB** este o distribuție de *Apache Derby* suportată de compania Sun.
- **Java DB** este un server complet tranzacțional, sigur, bazat pe standarde de baze de date, scris în întregime în Java, și care suportă în totalitate SQL, JDBC API și tehnologia Java EE.
- Baza de date **Java DB** este asamblată împreună cu serverul de aplicații *GlassFish* și este inclusă, de asemenea, în JDK 6.
- Dacă există serverul *GlassFish* înregistrat în instalarea NetBeans IDE, atunci **Java DB** va putea fi deja utilizată.

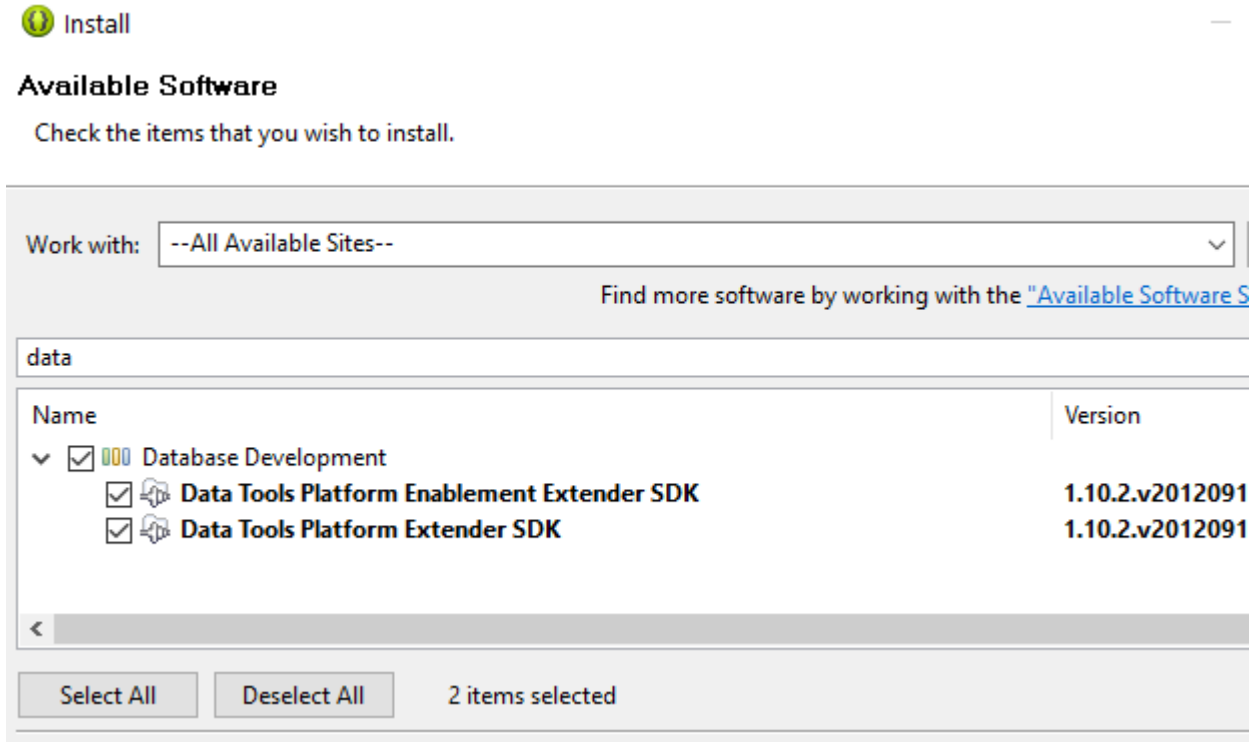
Lucrul cu baze de date în Java

- **NetBeans – Services – Databases**
- Crearea bazelor de date, tabelelor și popularea acestora



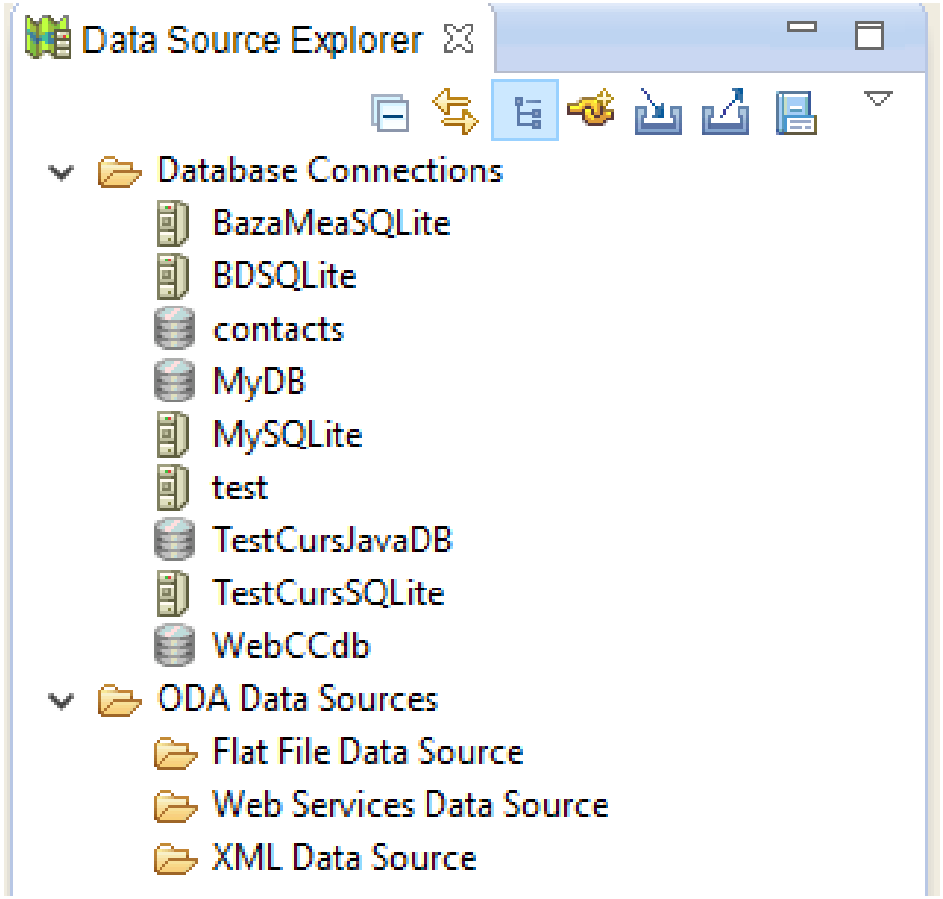
Lucrul cu baze de date în Java

- **Eclipse - Install New Software - Database Development**
- Crearea bazelor de date, tabelelor și popularea acestora



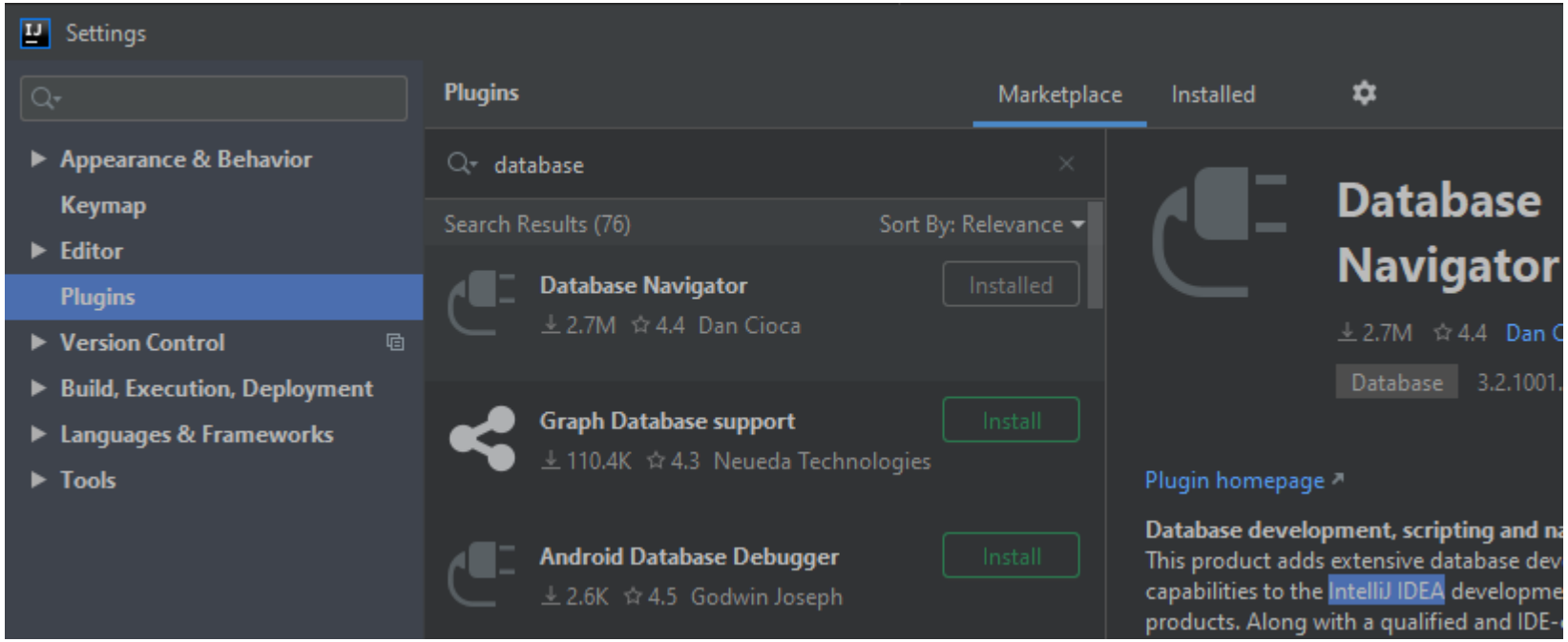
Lucrul cu baze de date în Java

- **Eclipse - Install New Software - Database Development**
- Crearea bazelor de date, tabelelor și popularea acestora



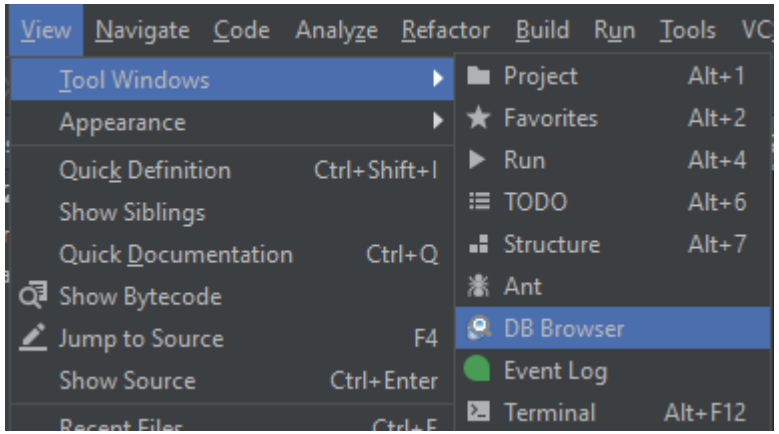
Lucrul cu baze de date în Java

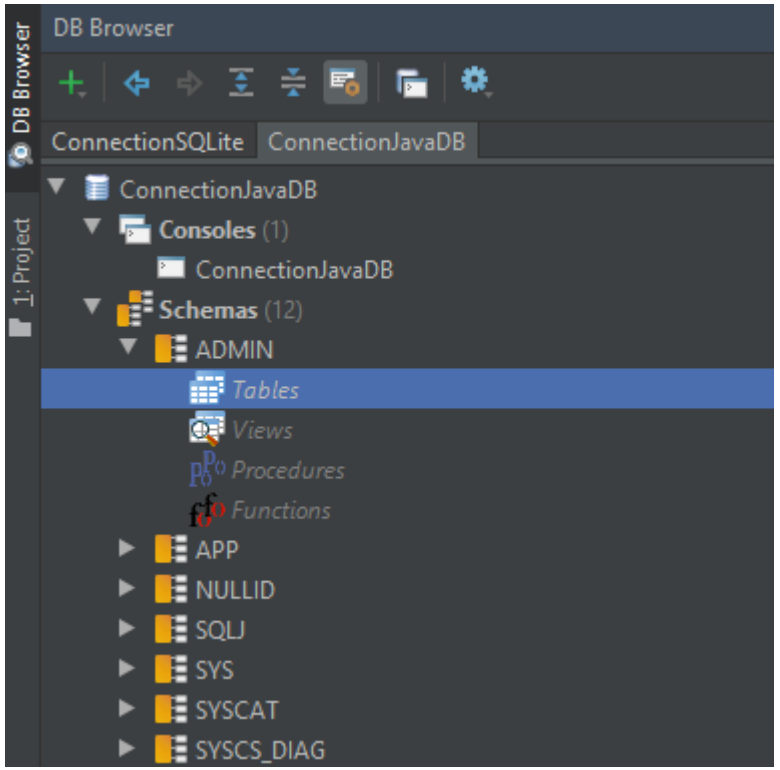
- **IntelliJ IDEA – Settings – Plugins**



Lucrul cu baze de date în Java

- **IntelliJ IDEA – DB Browser**





Lucrul cu baze de date în Java

În fereastra de *Servicii* a IDE-ului **NetBeans** sau din perspectiva *Database Development* a IDE-ului **Eclipse** se pot realiza următoarele operații asupra structurii bazei de date:

- crearea, ștergerea, modificarea tabelelor;
- popularea tabelelor cu date;
- vizualizarea datelor în format tabelar;
- execuția de instrucțiuni SQL și interogări.

Lucrul cu baze de date în Java

Interfețele și clasele pentru JDBC se găsesc în pachetul **java.sql.***.

Procesarea instrucțiunilor SQL cu JDBC presupune:

- stabilirea unei conexiuni;
- crearea unei instrucțiuni;
- execuția unei interogări;
- prelucrarea obiectului **ResultSet**;
- închiderea conexiunii.

Lucrul cu baze de date în Java

- Se apelează clasa **DriverManager** cerând un driver specific pentru baza de date;
- Driverul specific creează legătura cu baza de date și returnează un obiect de tip **Connection**;
- Cu ajutorul obiectului de tip **Connection** se creează un obiect **Statement** care conține și o cerere SQL către baza de date;
- Obiectul **Statement** returnează rezultatele într-un obiect **ResultSet**.

Lucrul cu baze de date în Java

- Crearea unei conexiuni la o bază de date:
- `Connection conn = DriverManager.getConnection (host, username, password);`

- Execuția unei instrucțiuni SQL asupra unei tabele necesită crearea unui obiect **Statement**:
- `import java.sql.Statement;`
- `Statement stmt = con.createStatement( );`

Lucrul cu baze de date în Java

Metodele cele mai importante ale obiectului *Statement*:

- *executeQuery(String)* – execută comanda SQL, returnează un obiect de tip *ResultSet* și se utilizează pentru execuția comenzilor SELECT;
- *executeUpdate(String)* – execută comanda SQL primită ca parametru și returnează numărul rândurilor tabelei modificate. Se utilizează pentru comenzile SQL de manipulare a datelor (INSERT, UPDATE, DELETE) și pentru comenzi de definire a datelor (CREATE/DROP TABLE);
- *execute(String)* – poate fi privită ca fiind generalizarea celorlaltor două metode. Se utilizează dacă comanda SQL poate returna deodată mai multe rezultate sau nu se cunoaște rezultatul execuției.

Lucrul cu baze de date în Java

Instrucțiunile JDBC se reprezintă utilizând următoarele clase:

- **Statement** - instrucțiunea este trimisă la serverul de bază de date de fiecare dată;
- **PreparedStatement** - instrucțiunea este stocată în memoria cache și apoi calea de execuție este predeterminată pe serverul bazei de date, permițându-i să fie executată de mai multe ori într-o manieră eficientă;
- **CallableStatement** - folosit pentru executarea procedurilor stocate în baza de date.

Operații DDL/DML

- Crearea unei tabele (CREATE TABLE);
- Operația de inserare date (INSERT);
- Operația de interogare (SELECT);
- Operația de actualizare (UPDATE);
- Operația de ștergere (DELETE).

Operații DDL/DML

- `private connect()`
- `public boolean update()`
- `public ResultSet query(String strSql) {`

```
try {
  Statement tmpStatement =
  connection.createStatement();
  ResultSet resultSet =
  tmpStatement.executeQuery(strSql);
  return resultSet;
  }
  catch (SQLException ex) {
    //handle exception here
    return null; }
}
```

Operații DDL/DML

- Un **ResultSet** este o modalitate de a stoca și manipula înregistrările returnate dintr-o interogare SQL.
- Odată ce avem la dispoziție toate înregistrările într-un **ResultSet**, există metode pe care le putem utiliza pentru a manipula înregistrările:
  - **`next()`** – poziționare pe linia următoare;
  - **`previous()`** – poziționare pe linia precedentă;
  - **`first()`** – poziționare pe prima linie;
  - **`last()`** – poziționare pe ultima linie;
  - **`absolute()`** – poziționare pe poziția indicată.

Operații DDL/DML

- **ResultSetMetaData** este un obiect care poate fi folosit pentru a obține informații despre tipurile și proprietățile coloanelor dintr-un obiect **ResultSet**:
  - **getColumnCount();**
  - **getColumnName(i);**
  - **getColumnTypeName(i);**

Lucrul cu baze de date în Java

- În Java nu există niciun cadru de interogare care să ofere o integrare directă la nivel de limbaj așa cum face LINQ pentru C# (similar cu Entity Framework).
- Alternative Java la Entity Framework :
  - **JPA (Java Persistence API)**: un ORM standard Java pentru stocarea, accesul și gestiunea obiectelor Java într-o bază de date relațională. Este parte integrantă a Java Enterprise Edition;
  - **Hibernate**: instrument de mapare obiect-relațional pentru limbajul de programare Java. Acesta oferă un cadru pentru maparea unui model orientat pe obiecte la o bază de date relațională.

Bibliografie

- [1] Jonathan Knudsen, Patrick Niemeyer – *Learning Java, 3rd Edition*, O'Reilly.
- [2] http://www.itcsolutions.eu
- [3] http://www.acs.ase.ro
- [4] http://www.tutorialspoint.com/sqlite/sqlite_java.htm
- [5] https://netbeans.org/kb/docs/ide/java-db.html
- [6] https://www.ms.sapientia.ro/~manyi/teaching/oop/oop_romanian/curs10/curs10.html
- [7] http://www.vogella.com/tutorials/EclipseDataToolsPlatform/article.html