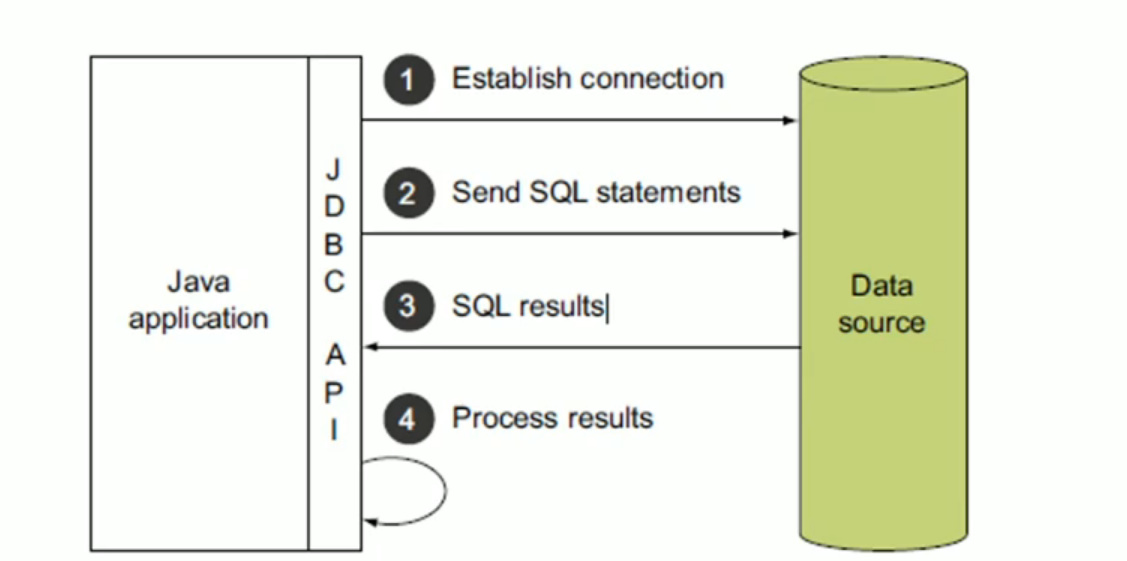
JDBC – standart care conecteaza o aplicatie Java cu un RDBMS



**Clase in java.sql**

* Driver – asigura conectarea la baza de date. Adica, ofera posibilitatea de a crea conexiuni la baza de date
* Connection
* Statement – simpla clase care ne ajuta sa executam query. Ne ajuta sa extragem date in baza de date
* PreparedStatement – ne ajuta sa inseram date in baza de date si previne sql injection.
* CallableStatement – apeleaza functii si proceduri din baza de date
* ResultSet – pentru datele care vin inapoi

**Connection Method**

* connection.setAutoCommit(false/true) – default e true. Orice operatie in baza de date va fi automat commited, deci nu va fi necesar sa apelam .commit()
* connection.commit()
* connection.rollback()
* connection.execute(StringQuery)
* connection.updateExecute(StringQuery)
* connection.executeQuery(StringQuery)

**Examples**

public static void main(String[] args) throws SQLException {  
 MysqlDataSource dataSource = new MysqlDataSource();  
 dataSource.setUser("testuser");  
 dataSource.setPassword("Frb2eshox!");  
 dataSource.setURL("jdbc:mysql://localhost:3306/test");  
  
 Connection connection = dataSource.getConnection();  
  
 Statement statement = connection.createStatement();  
 statement.execute("INSERT INTO teacher VALUES(100,'Cornel','Catan',18)");  
 connection.close();  
  
}

**Statement**

public static void main(String[] args) throws SQLException {  
 MysqlDataSource dataSource = new MysqlDataSource();  
 dataSource.setUser("testuser");  
 dataSource.setPassword("Frb2eshox!");  
 dataSource.setURL("jdbc:mysql://localhost:3306/test");  
  
 Connection connection = dataSource.getConnection();  
  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery("SELECT \* FROM teacher");  
 resultSet.next();  
 String teacher = String.*format*("Teacher = {id=%d, first\_name=%s, last\_name=%s, expirience=%d}",  
 resultSet.getInt("id"),resultSet.getString("first\_name"),resultSet.getString("last\_name"),resultSet.getInt("expirience"));  
 System.*out*.println(teacher);  
  
 connection.close();  
  
}

**Statment cu mai multe rezultate**

public static void main(String[] args) throws SQLException {  
 MysqlDataSource dataSource = new MysqlDataSource();  
 dataSource.setUser("testuser");  
 dataSource.setPassword("Frb2eshox!");  
 dataSource.setURL("jdbc:mysql://localhost:3306/test");  
  
 Connection connection = dataSource.getConnection();  
  
 Statement statement = connection.createStatement();  
 String query = "SELECT \* FROM teacher";  
 statement.execute(query);  
 ResultSet resultSet = statement.getResultSet();  
  
 while(resultSet.next()){  
 System.*out*.printf("Teacher = {%d, %s, %s, %d}",resultSet.getInt("id"),resultSet.getString("first\_name"),resultSet.getString("last\_name"),resultSet.getInt("expirience"));  
 }  
   
 connection.close();  
  
}

**PreparedStatement**

MysqlDataSource dataSource = new MysqlDataSource();  
dataSource.setUser("testuser");  
dataSource.setPassword("Frb2eshox!");  
dataSource.setURL("jdbc:mysql://localhost:3306/test");  
  
Connection connection = dataSource.getConnection();  
  
PreparedStatement statement = connection.prepareStatement("INSERT INTO teacher VALUES(?,?,?,?)");  
statement.setInt(1,123);  
statement.setString(2,"CAPATINA");  
statement.setString(3,"VLADA");  
statement.setInt(4,25);  
statement.execute();  
  
  
connection.close();

**Observam ca avem ?, nu ?1**

**Connection Pool**

* O connection pool creaza un numar de conexiuni deja definite. La rularea programului se vor crea atatea conexiuni deodata, cate am specificat si anume aceste conexiuni vor fi mereu folosite
* In caz ca sunt folosite toate conexiunile, va fi necesar de a se crea una noua
* Cand o conexiune termina lucrul, ea e rentoarsa inapoi in pool
* cand sunt create un numar maxim de conexiuni si mai e nevoie de una pentru o transactie, transactia va astepta ca o conexiune sa se elibereze, adica sa fie inchisa si returnata in pool