JPA – Java Persistence API

JPA oferă facilităţi ORM (Object/Relational Mapping) pentru a gestiona date relaţionate în aplicaţii Java, în principal pentru a memora datele pe un suport persistent (de obicei bază de date relaţională) şi pentru a efectua interogări asupra datelor persistate. Java Persistence cuprinde:

- *Java Persistence API.*
- *Limbajul de interogare.*
- *Java Persistence Criteria API.*
- *Metadate de mapare.*

Unitatea elementară care se persistă se numeşte *entitate*. De obicei o entitate este asociată cu un tabel din baza de date, o linie (înregistrare) din tabel fiind o instanţă a entităţii. O entitate are câmpuri sau proprietăţi persistente, care sunt asociate prin mapări cu elementele specifice suportului de stocare.

O clasă trebuie să îndeplinească anumite condiţii pentru a fi entitate, ea va fi în plus marcată cu adnotarea *@Entity*.

Se pot impune anumite validări prin adnotări specifice. Astfel, un câmp obligatoriu se marchează cu *@NotNull* iar un format particular de introducere este precizat prin *@Pattern(regexp="...")*.

Entităţile posedă elemente unice de identificare numite *chei primare*, care pot fi simple (un singur câmp) sau compuse (mai multe câmpuri).

Relaţiile dintre entităţi pot avea următoarele *multiplicităţi*:

- **One-to-one**, în care o instanţă a unei entităţi este legată (relaţionată) cu o singură instanţă a altei entităţi.
- **One-to-many**, în care o instanţă a unei entităţi poate fi legată (relaţionată) la mai multe instanţe ale celeilalte entităţi. Exemplul clasic este cel al unei comenzi (order) care poate avea mai multe linii.
- **Many-to-one**, este relaţia inversă faţă de one-to-many, când mai multe instanţe ale unei entităţi pot fi legate (relaţionate) la o instanţă a celeilalte entităţi.
- **Many-to-many**, când instanţele unei entităţi pot fi legate la mai multe instanţe ale celeilalte entităţi. De exemplu, un curs are mai mulţi studenţi şi fiecare student urmează mai multe cursuri.

Relaţiile pot fi:

- **Bidirecţionale**, în care fiecare entitate are un câmp prin care se referă la cealaltă entitate (fiecare entitate "ştie" de cealaltă).
- **Unidirecţionale**, în care doar o entitate "ştie" de cealaltă.

Aceste relaţii crează anumite dependenţe, care impun anumite reguli la realizarea anumitor operaţii (ştergere).

Entităţile sunt gestionate de un manager de entităţi, o instanţă a **javax.persistence.EntityManager**, care este asociat cu un context de persistenţă (o mulţime de entităţi care există într-un anumit depozit de informaţii).

Instanţele de entităţi pot fi în una din stările următoare:

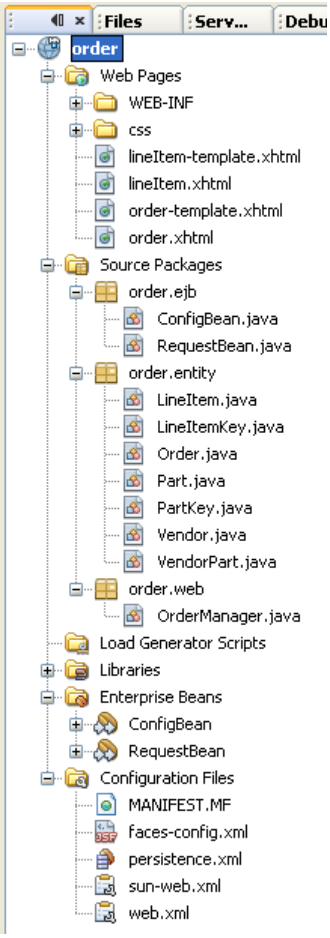
- *New*
- *Managed*
- *Detached*
- *Removed*

1. Aplicația "order"

Deschideţi în NetBeans proiectul "***order***", care se găseşte în tutorialul JEE6 (examples/persistence) sau în arhiva cu surse de pe site. Aceasta este o aplicaţie simplă de gestiune a unor produse şi de realizare de comenzi pe aceste produse.

Analizaţi fişierele componente şi modul lor de organizare.

Identificaţi entităţile aplicaţiei (produs, furnizor, comandă, linie de comandă).

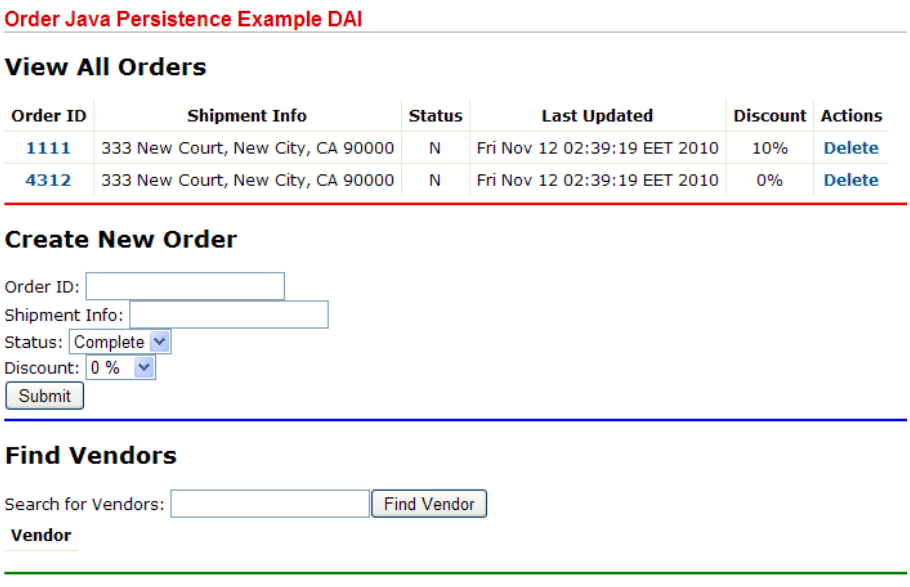


Identificaţi relaţiile dintre entităţi şi multiplicităţile acestora.

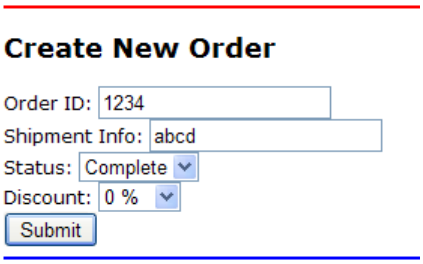
Rulaţi proiectul din NetBeans, identificând pagina de pornire (în fişierul de configurare).

Identificaţi şi analizaţi cele 2 şabloane folosite.

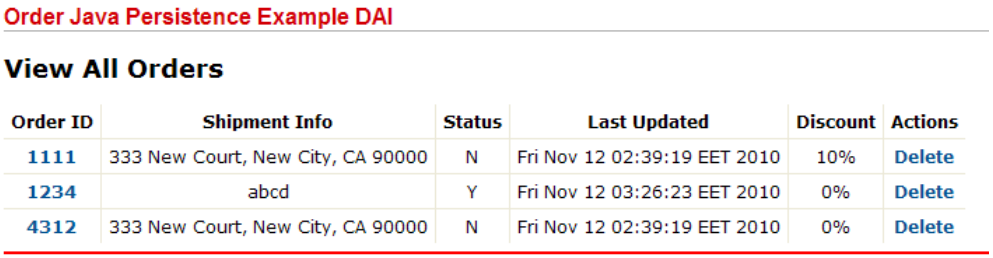
Aplicaţi modificările necesare pentru a schimba aspectul implicit al paginii de plecare:



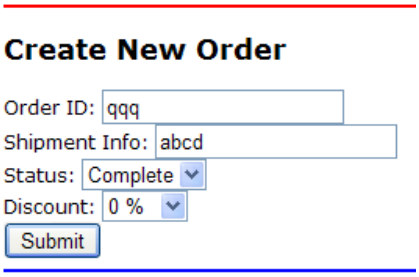
Adăugaţi o comandă nouă, completând valori corecte în câmpurile comenzii:



Observaţi efectul imediat:



Adăugați o comandă nouă, completând valori incorecte în câmpurile comenzii:

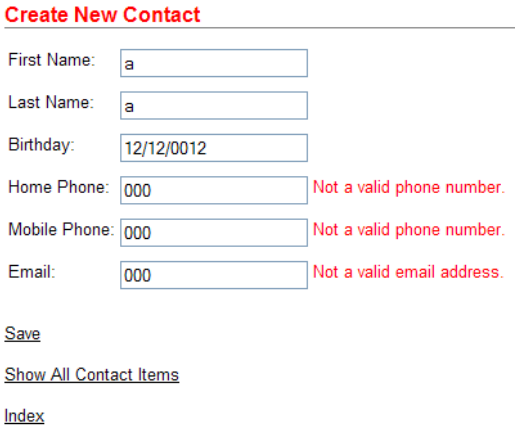


Identificați de ce nu se adaugă înregistrarea dorită. Verificați unde se semnalează problemele.
Identificați cum se face "încărcarea" inițială a datelor.
Adăugați un buton de ștergere produs de pe comandă.
Adăugați un atribut nou pe comandă, numărul de produse conținute. Acesta va fi (re)calculat automat la adăugarea sau ștergerea unui produs.

2. Aplicația "address-book"

Deschideţi în NetBeans proiectul "address-book", care se găseşte în tutorialul JEE6 (examples/persistence) sau în arhiva cu surse de pe site. Aceasta este o aplicaţie simplă de gestiune a unei liste de contacte.

Analizaţi fişierele componente şi modul lor de organizare.
Identificaţi entităţile aplicaţiei (contact).
Modificaţi mesajele afişate la introducerea incorectă a datelor.



Adăugaţi un câmp nou pe entitate, care să memoreze ocupaţia persoanei şi propagaţi acest câmp în toate paginile necesare.

3. Aplicația "criteriaQuery"*

Deschideți în NetBeans proiectul "criteriaQuery", care se găseşte în exemplele JPA (https://glassfish-samples.dev.java.net/servlets/ProjectDocumentList?folderID=5214&expandFolder=5214&folderID=0) sau în arhiva cu surse de pe site. Descrierea acestei aplicaţii se găseşte la adresa https://glassfish-samples.dev.java.net/source/browse/*checkout*/glassfish-samples/tags/JAVAEE6_SAMPLES-0_9-b16/ws/javaee6/jpa/criteriaQuery/docs/index.html.

Identificaţi entităţile acestei aplicaţii.

Rulaţi aplicaţia, mai întâi direct, apoi pas cu pas (depanare).

4. Resurse JPA

Resurse utile:

- http://download.oracle.com/javaee/6/tutorial/doc/
- http://download.oracle.com/javaee/6/tutorial/doc/bnbpy.html
- http://download.oracle.com/javaee/6/javaserverfaces/2.0/docs/pdldocs/facelets/h/tld-summary.html
- http://www.coreservlets.com/JSF-Tutorial/jsf1/
- http://www.coreservlets.com/JSF-Tutorial/jsf2/
- https://javaeetutorial.dev.java.net/servlets/ProjectDocumentList
- http://download.oracle.com/javaee/6/javaserverfaces/2.0/docs/pdldocs/facelets/
- http://netbeans.org/kb/docs/web/jsf20-intro.html

JDBC – Java Database Connectivity

JDBC reprezintă un API pentru realizarea operaţiilor cu baze de date (relaţionale) din aplicaţii Java. Clasele JDBC se găsesc în pachetul java.sql, existând şi o extensie în javax.sql.
În general prelucrarea unei operaţii SQL cu JDBC urmează o succesiune de paşi:

- Stabilirea unei conexiuni la sursa de date (bază de date, sistem de fişiere etc).
- Crearea unei fraze SQL (un obiect Statement se crează cu ajutorului unui obiect Connection).
- Executarea interogării (regăsire date, apel procedură stocată, modificare date).
- Prelucrarea obiectului `ResultSet` rezultat.
- Închiderea conexiunii, pentru eliberarea resurselor folosite.

Există 3 feluri de fraze SQL:

- `Statement`: fraze SQL simple, fără parametri.
- `PreparedStatement`: extind `Statement` şi pot conţine parametri de intrare.
- `CallableStatement`: extend `PreparedStatement` şi se folosesc pentru a executa proceduri stocate ce pot avea atât parametri de intrare cât şi parametri de ieşire.

Execuţia unei interogări se face prin apelul unor metode ale obiectului `Statement`:

- `execute`: poate returna unul sau mai multe obiecte `ResultSet`, care pot fi obţinute prin apelarea repetată a metodei `Statement.getResultSet`.
- `executeQuery`: returnează u obiect `ResultSet`.
- `executeUpdate`: returnează numărul de linii afectate de fraza SQL de tip `INSERT`, `DELETE`, `UPDATE`.

Operaţiile uzuale sunt descrise de următoarea secvenţă de cod:

```
try {
      stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery(query);
      while (rs.next()) {
        String coffeeName = rs.getString("COF_NAME");
        int supplierID = rs.getInt("SUP_ID");
        float price = rs.getFloat("PRICE");
        int sales = rs.getInt("SALES");
        int total = rs.getInt("TOTAL");
        System.out.println(coffeeName + "\t" + supplierID + "\t" + price + "\t" + sales +
                                        "\t" + total);
      }
```

Specificarea sursei de date se face prin indicarea unui şir de caractere de tip URL:

- Pentru MySQL:

```
jdbc:mysql://[host][,failoverhost...][:port]/[database][?propertyName1][=propertyValue1][&
propertyName2][=propertyValue2]...
```

  - unde host:port implicit 127.0.0.1:3306.

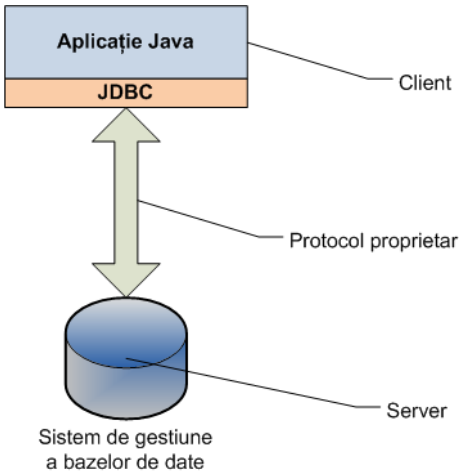
- Pentru Java DB:

```
jdbc:derby:[subsubprotocol:][databaseName][;attribute=value]*
```

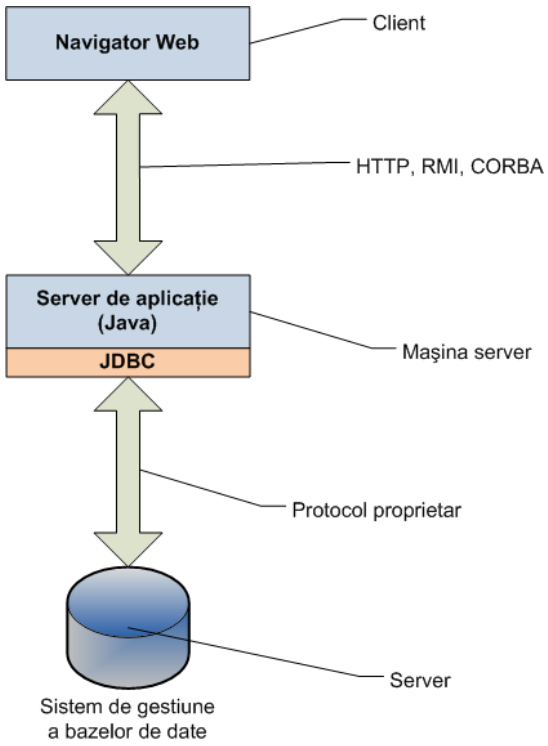
  - *subsubprotocol* (unde e baza de date: intr-un director, în memory, in class path, în JAR).
  - *databaseName* is the name of the database to connect to.
  - *attribute=value* listă opţională de atribute, pentru indicarea unor task-uri:

- Crearea bazei de date din URL.
- Criptarea bazei de date din URL.
- Director pentru log.
- Utilizator şi parolă de conectare.

Arhitectura JDBC în modelul cu 2 straturi este următoarea:



Arhitectura JDBC în modelul cu 3 straturi este următoarea:



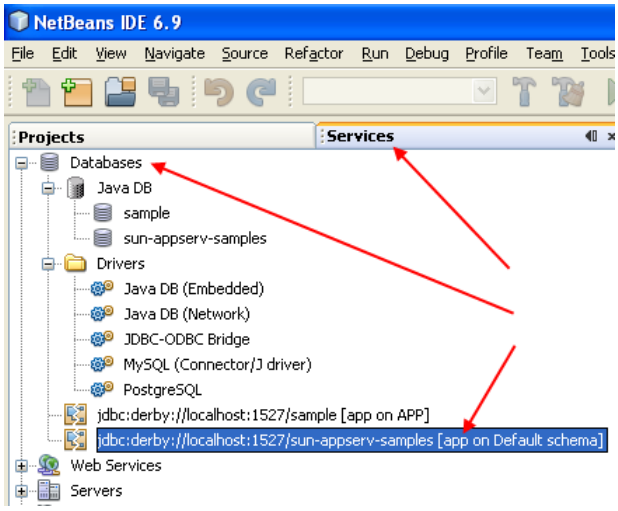
1. Derby (Java DB)

Derby este un motor de bază de date relaţională pur Java, care foloseşte standardul SQL şi JDBC ca API, oferind o instanţă încorporată în aplicaţie sau de server de reţea şi o serie de utilitare în linie de comandă:

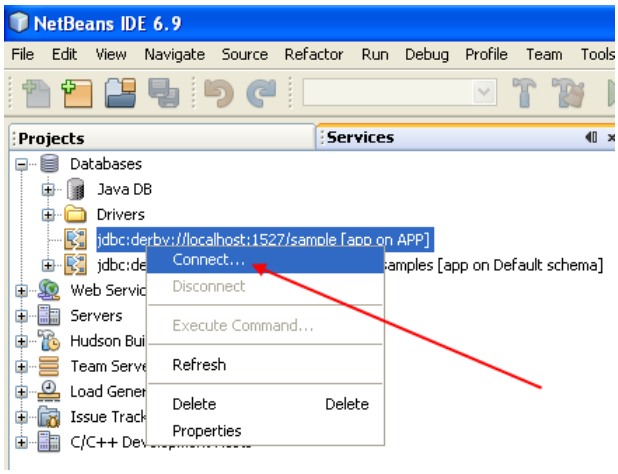
- ij (SQL scripting).
- dblook (schema dump).
- sysinfo (system info).

JDK 1.6 instalează un astfel de server în locaţia " C:\Program Files\Sun\JavaDB".

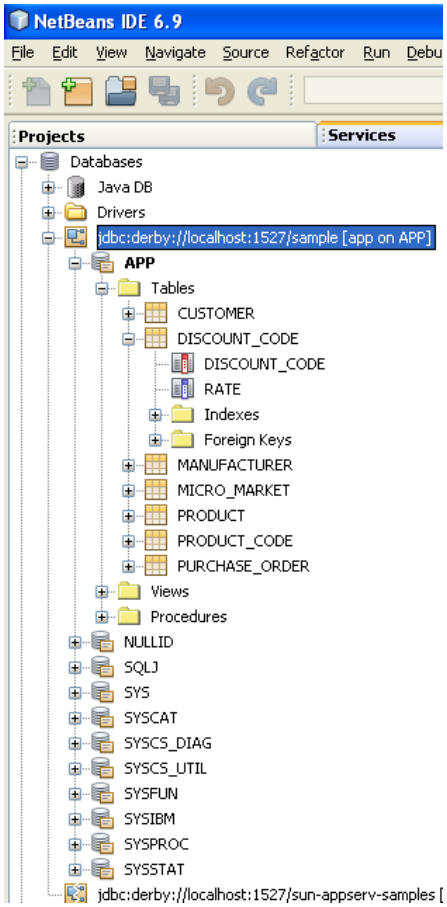
Mediul de dezvoltare NetBeans permite accesarea şi controlarea acestui server:



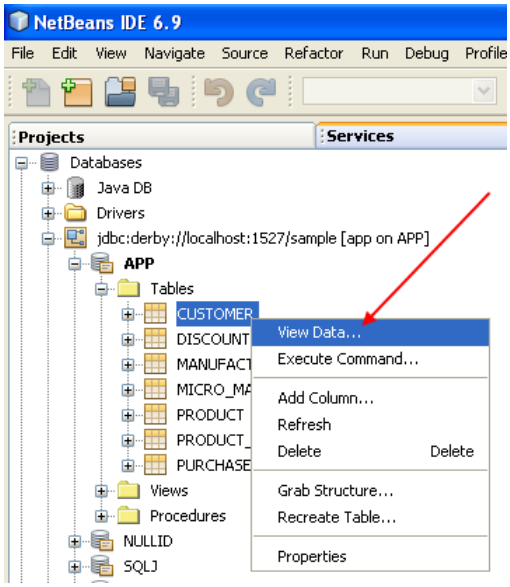
Startaţi manual acest server:



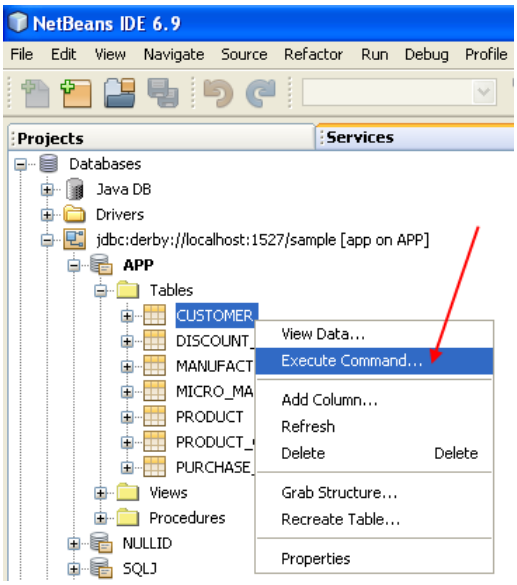
Analizaţi schemele care apar şi tabelele principale ale acestora:



Observaţi modalitatea de vizualizare a datelor din tabele şi analizaţi conţinutul acestora:

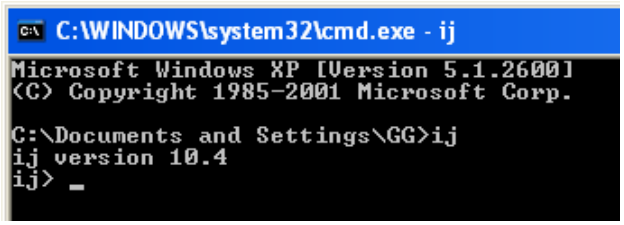


Executaţi manual o serie de interogări asupra datelor existente:

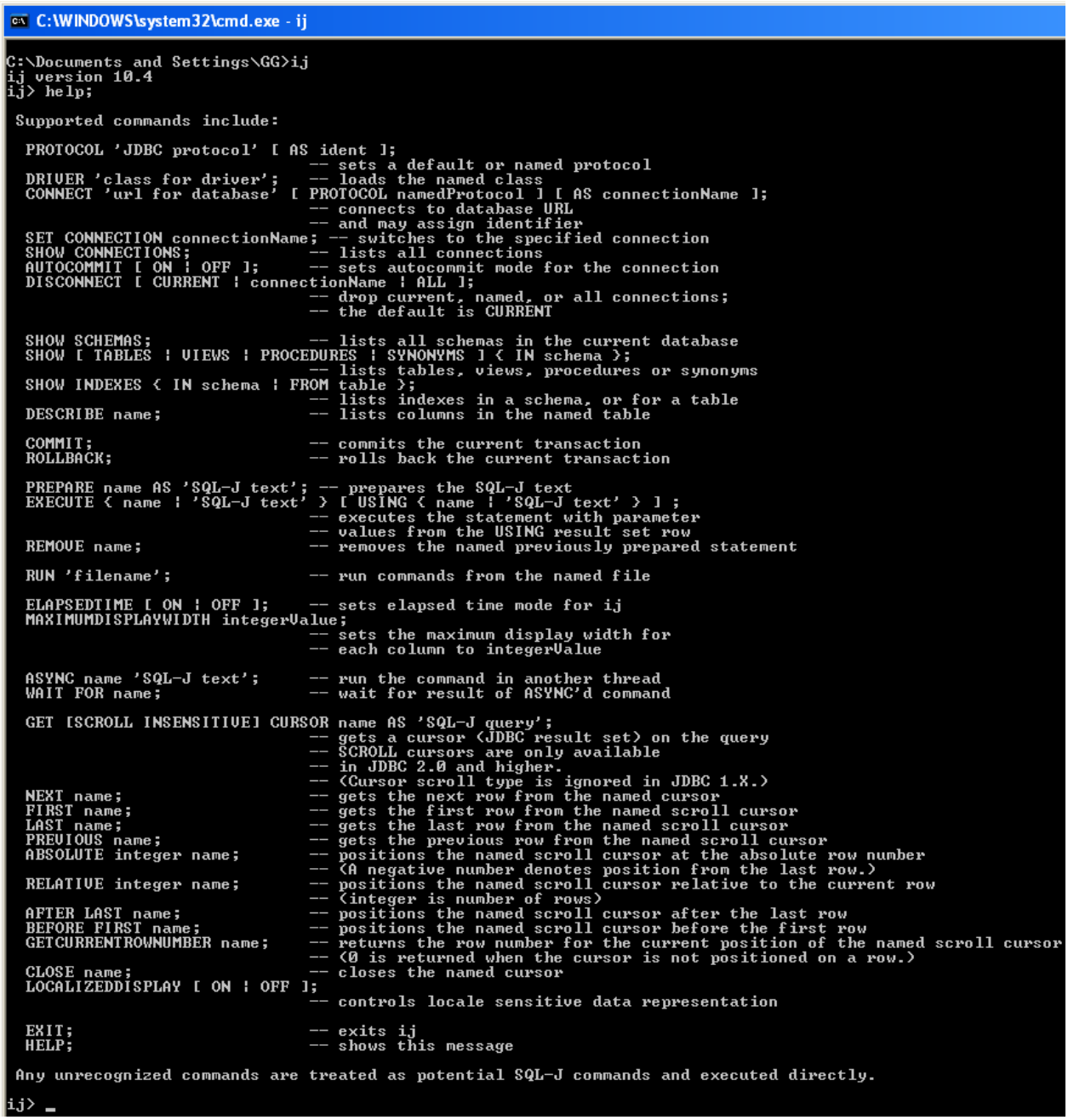


2. IJ*

Realizaţi setările de mediu necesare (variabilele DERBY_HOME, PATH) şi lansaţi utilitarul ij:

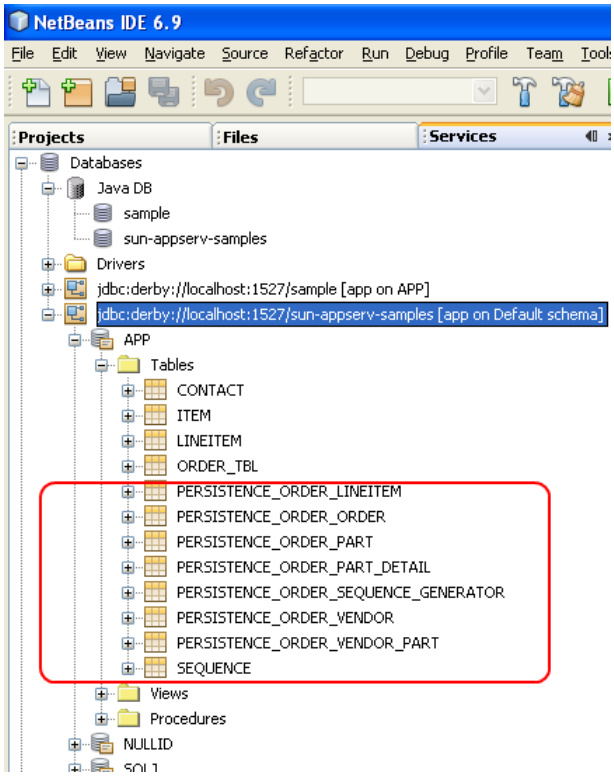


Comanda help oferă o serie de informaţii suplimentare:

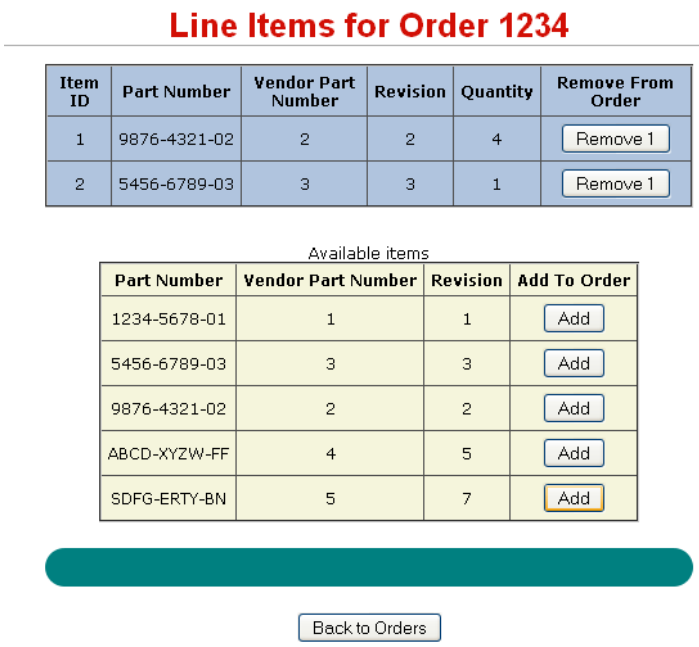


3. Aplicaţia "order"

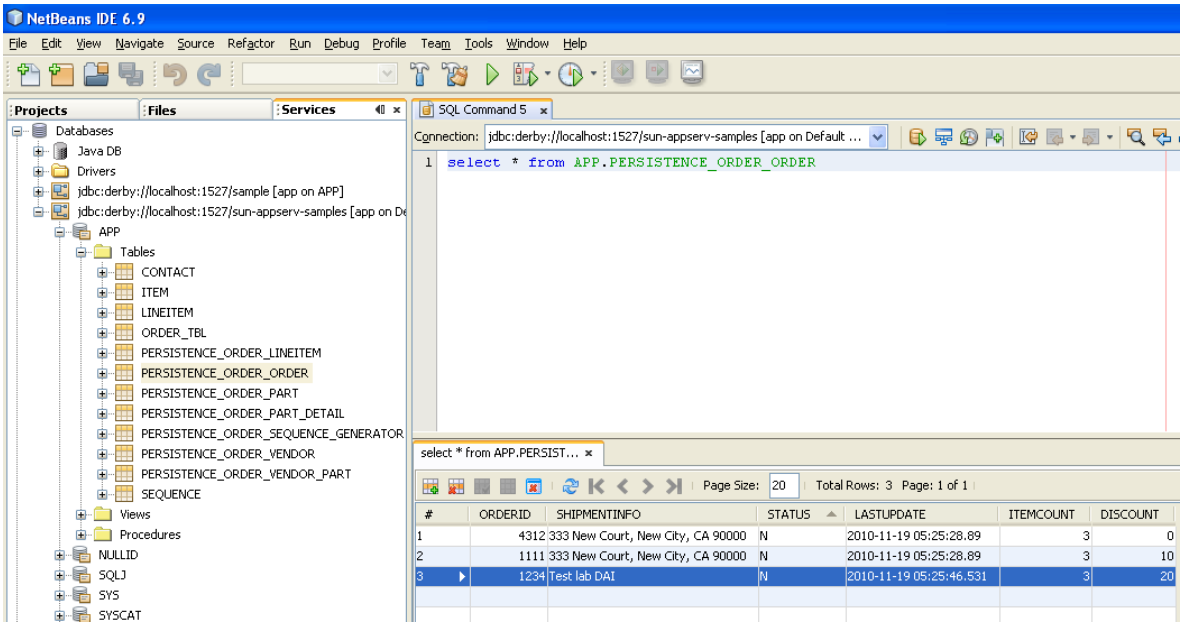
Deschideţi în NetBeans aplicaţia "order" folosită în secțiunea anterioară.
Identificaţi tabelele asociate entităţilor aplicaţiei.



Lansaţi aplicaţia şi adăugaţi comenzi şi produse.



Comparaţi cele afişate de aplicaţie cu informaţiile existente în baza de date.



4. Resurse JDBC

Resurse utile:

- http://download.oracle.com/javase/tutorial/jdbc/index.html
- http://download.oracle.com/javase/tutorial/jdbc/overview/index.html
- http://download.oracle.com/javase/tutorial/jdbc/basics/index.html
- http://www.oracle.com/technetwork/java/overview-141217.html
- http://www.jdbc-tutorial.com/
- http://www.herongyang.com/JDBC/
- http://onjava.com/pub/a/onjava/2006/08/02/jjdbc-4-enhancements-in-java-se-6.html
- file:///C:/Program%20Files/Sun/JavaDB/demo/programs/readme.html
- http://download.oracle.com/javaee/6/tutorial/doc/