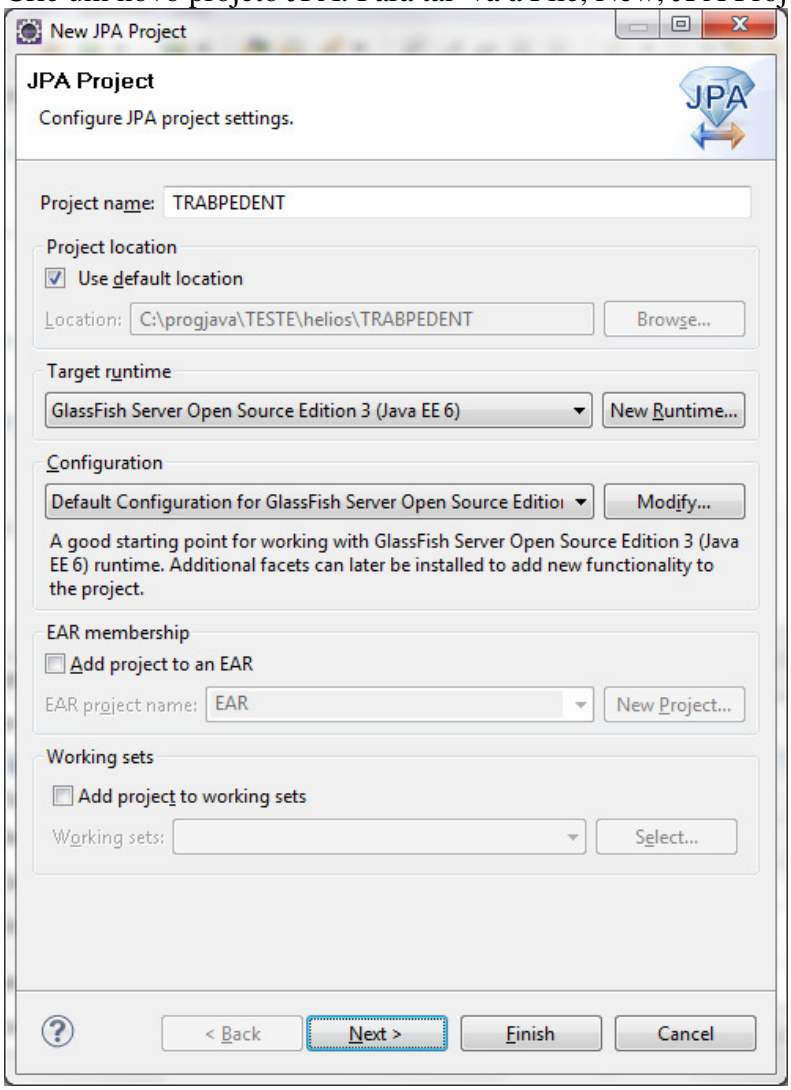
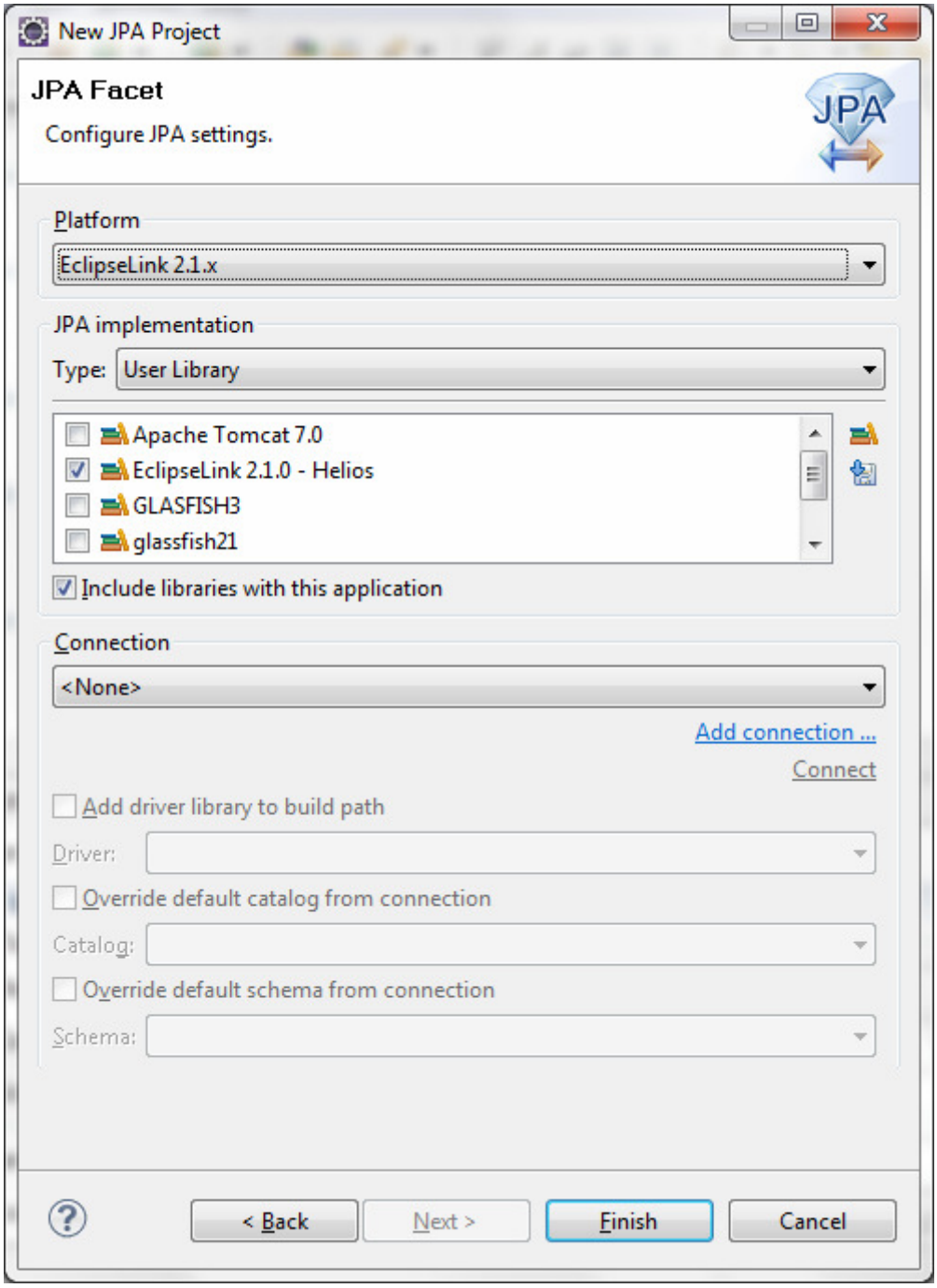
Tutorial JPA em ambiente Eclipse

Antes de iniciar execute o script do banco de dados no SQLYOG ou outra ferramenta de sua preferência

1-Selecione a perspectiva Java JPA: window, open perspective, other, JPA.
Crie um novo projeto JPA. Para tal vá a File, New, JPA Project.

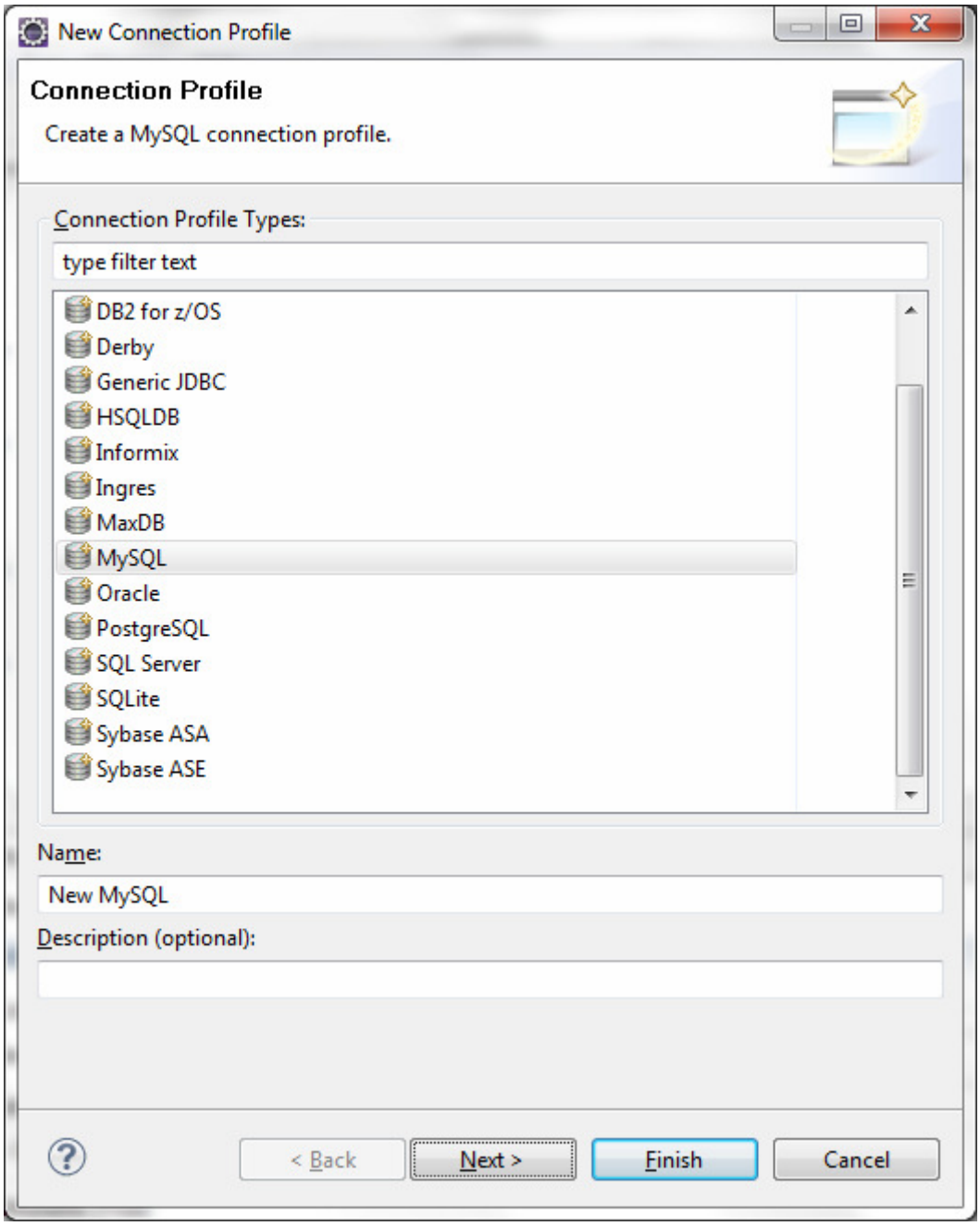


2-Defina a pasta onde o projeto será salvo e as outras opções deixe-as como na figura acima .
Clique next e na próxima tela, next novamente. A tela exibida será essa:

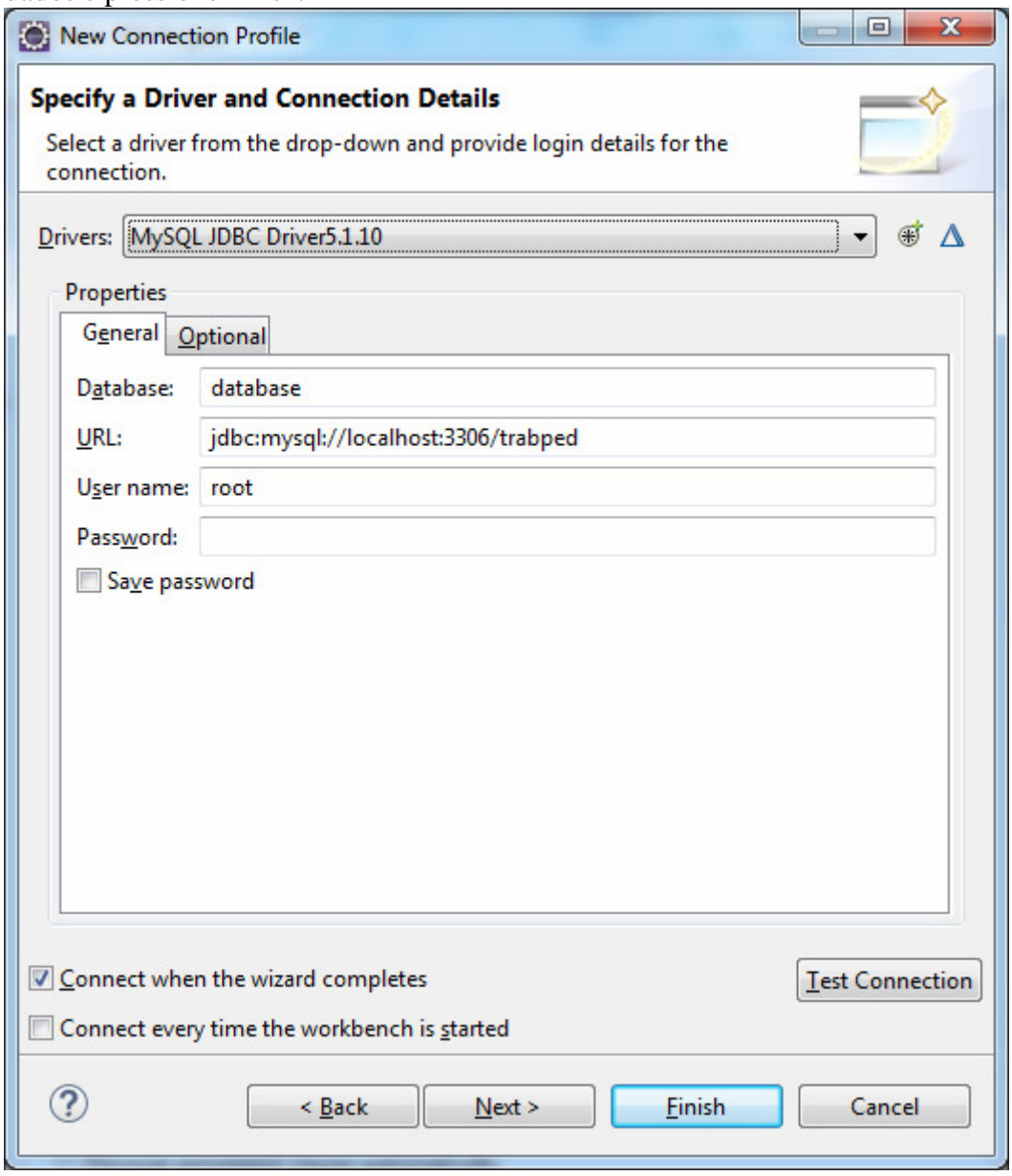


3-Se a plataforma EclipseLink não estiver disponível, clique no ícone  do lado direito da tela e siga as instruções para baixá-la. Após baixar, selecione a biblioteca EclipseLink conforme acima. Em *Connection,* se não houver conexões disponíveis, clique em *AddConnection*. Na tela exibida (abaixo), selecione a conexão Mysql e dê um

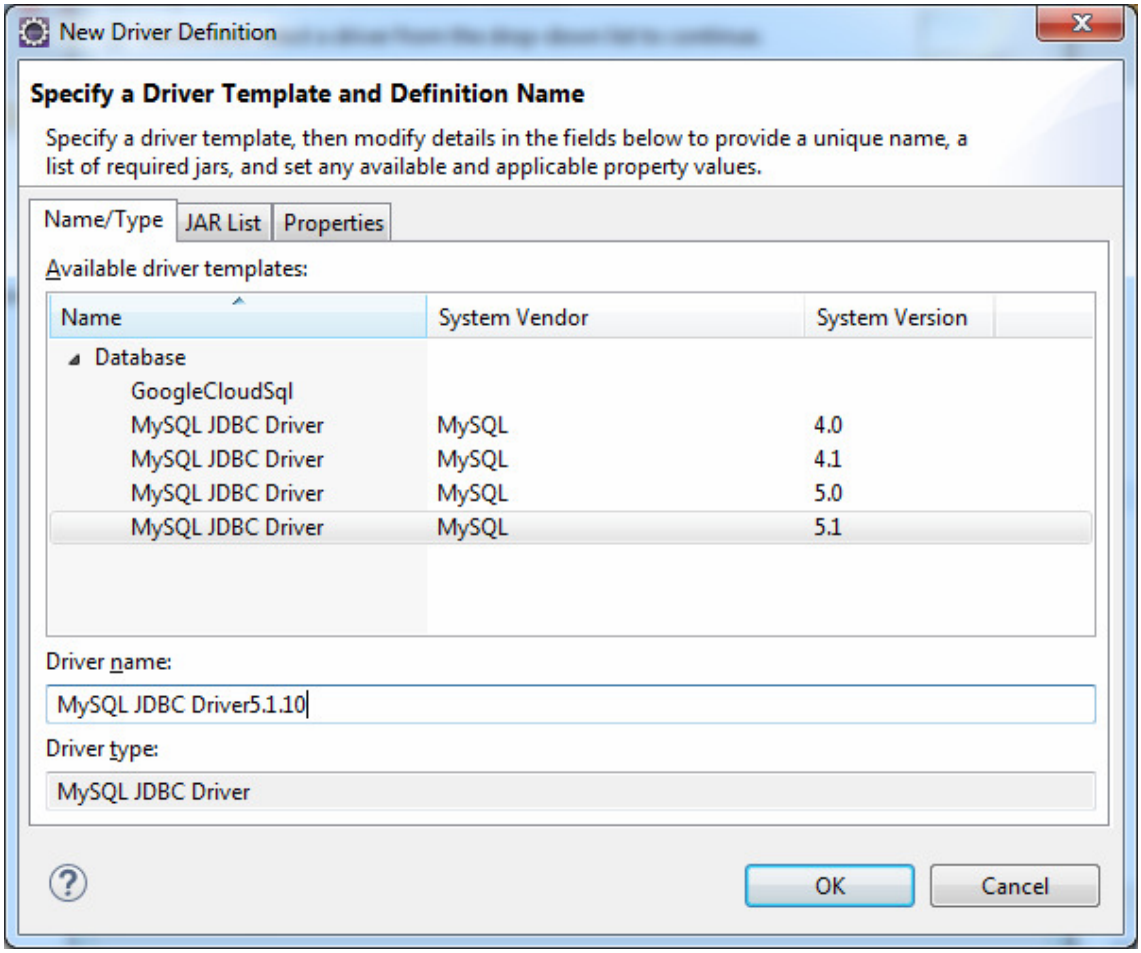
nome a ela.


New Connection Profile
Connection Profile
Create a MySQL connection profile.
Connection Profile Types:
type filter text
DB2 for z/OS
Derby
Generic JDBC
HSQLDB
Informix
Ingres
MaxDB
MySQL
Oracle
PostgreSQL
SQL Server
SQLite
Sybase ASA
Sybase ASE
Name:
New MySQL
Description (optional):
< Back
Next >
Finish
Cancel

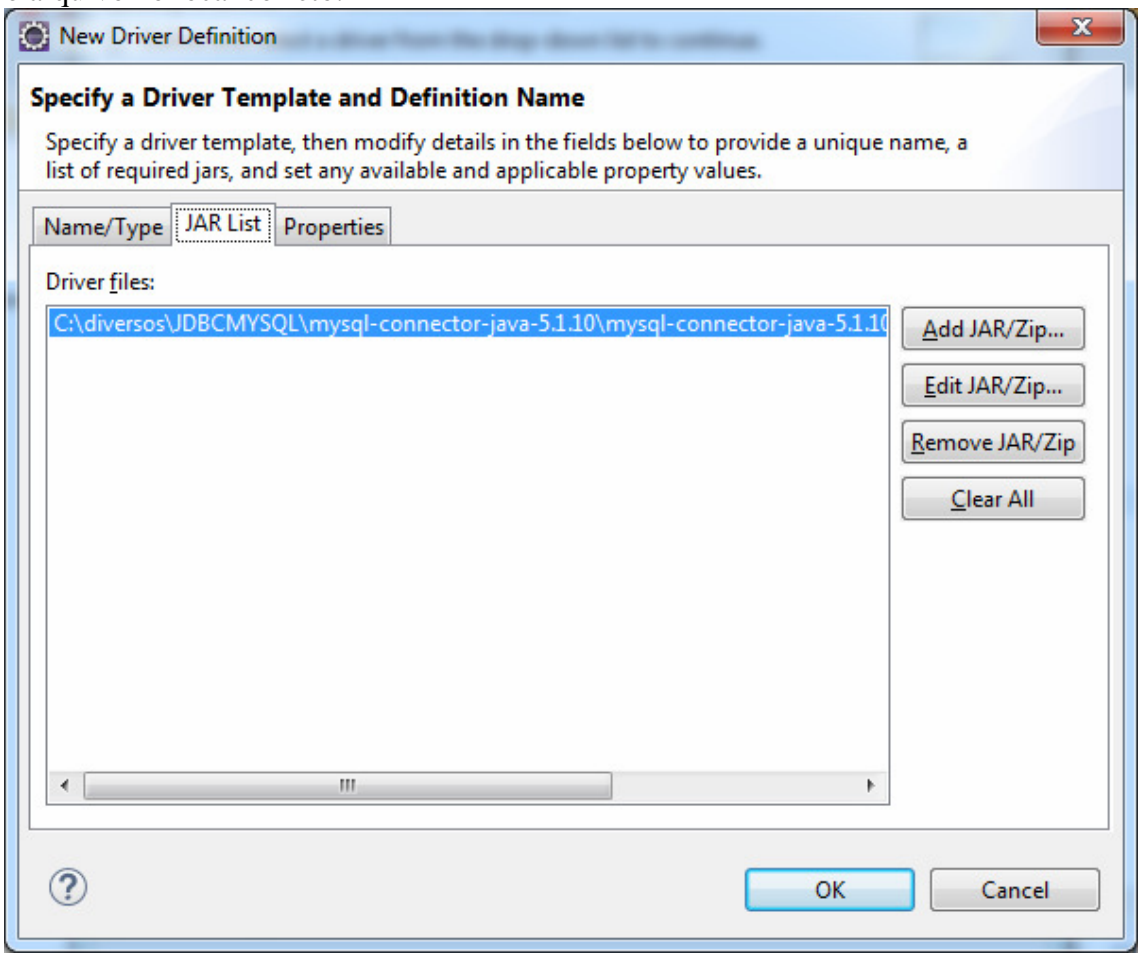
4-Depois pressione next e na tela seguinte (abaixo), preencha os dados do banco de dados e pressione finish.



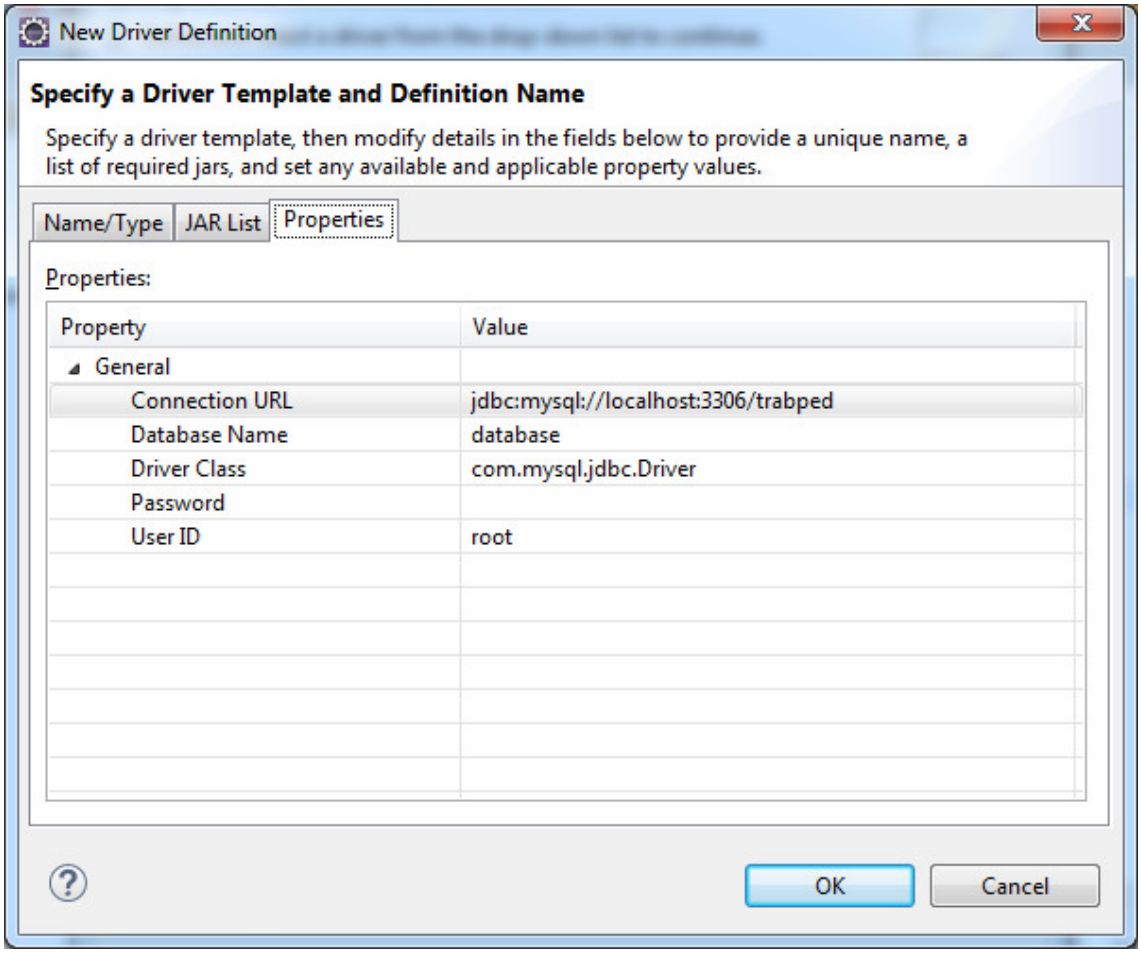
5-Se não houver nenhum driver no combobox *drivers* na tela acima, clique no ícone "+" à direita. A seguinte série de telas mostra o que deve ser selecionado.
5.1-Dê o nome mostrado em *DriverName:*



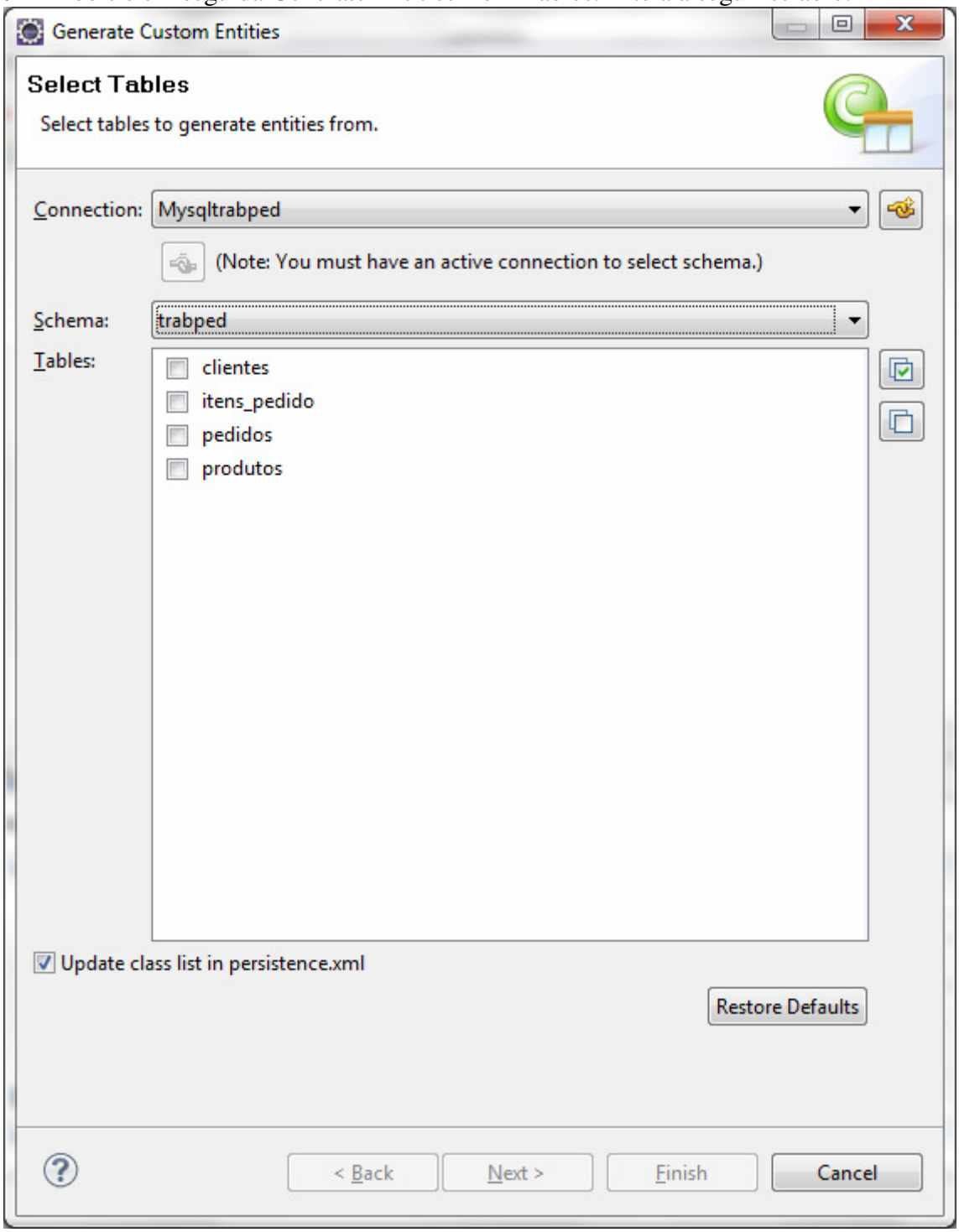
5.2-Clique na aba JAR List. Se não houver drivers, clique em Add JAR/Zip e selecione o arquivo no local correto:



5.3-Vá à aba Properties e preencha os dados do banco:

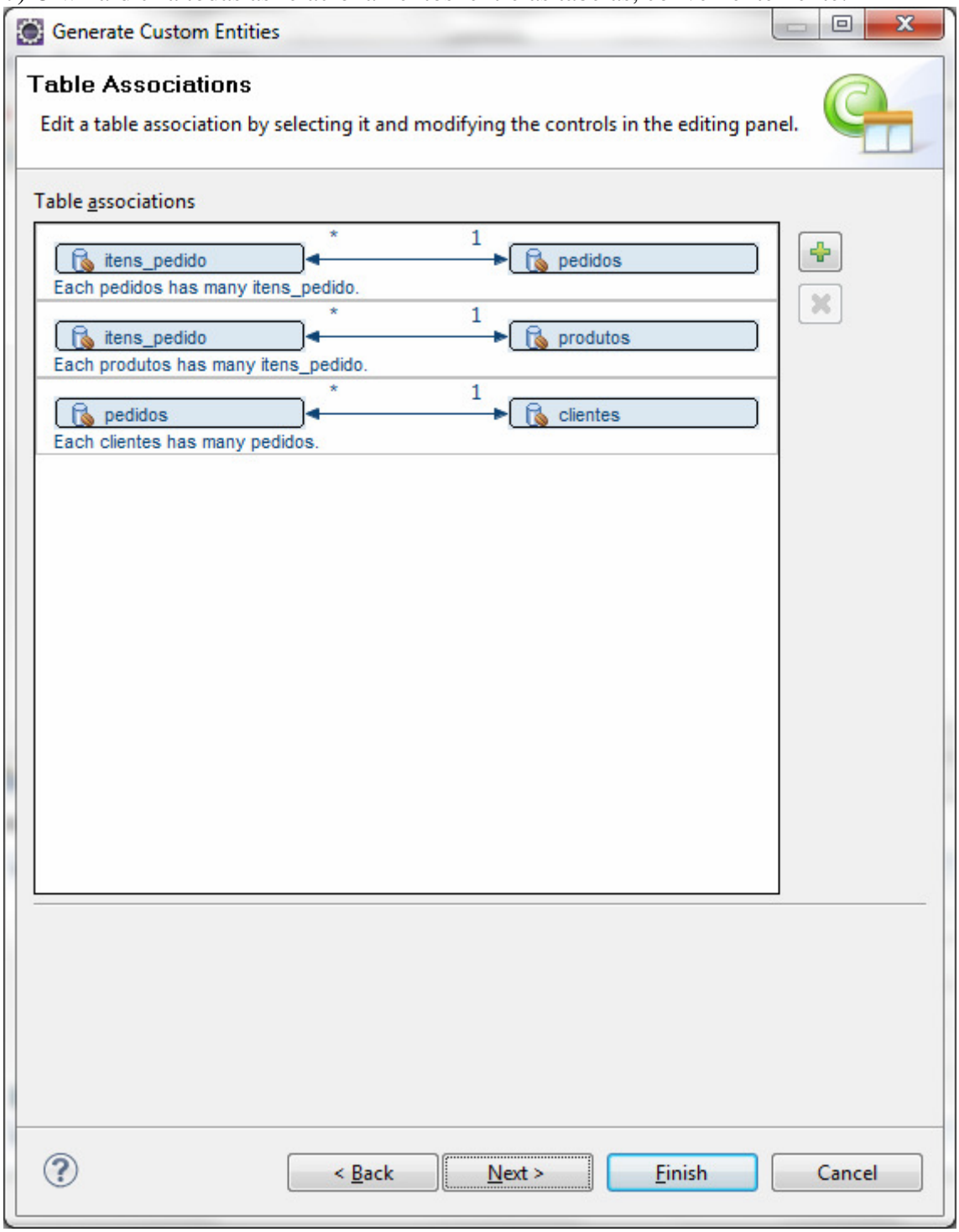


6-Criação das classes JPA. Clique com o botão direito sobre o nome do projeto , depois JPA Tools e em seguida Generate Entities from Tables. A tela a seguir se abre:



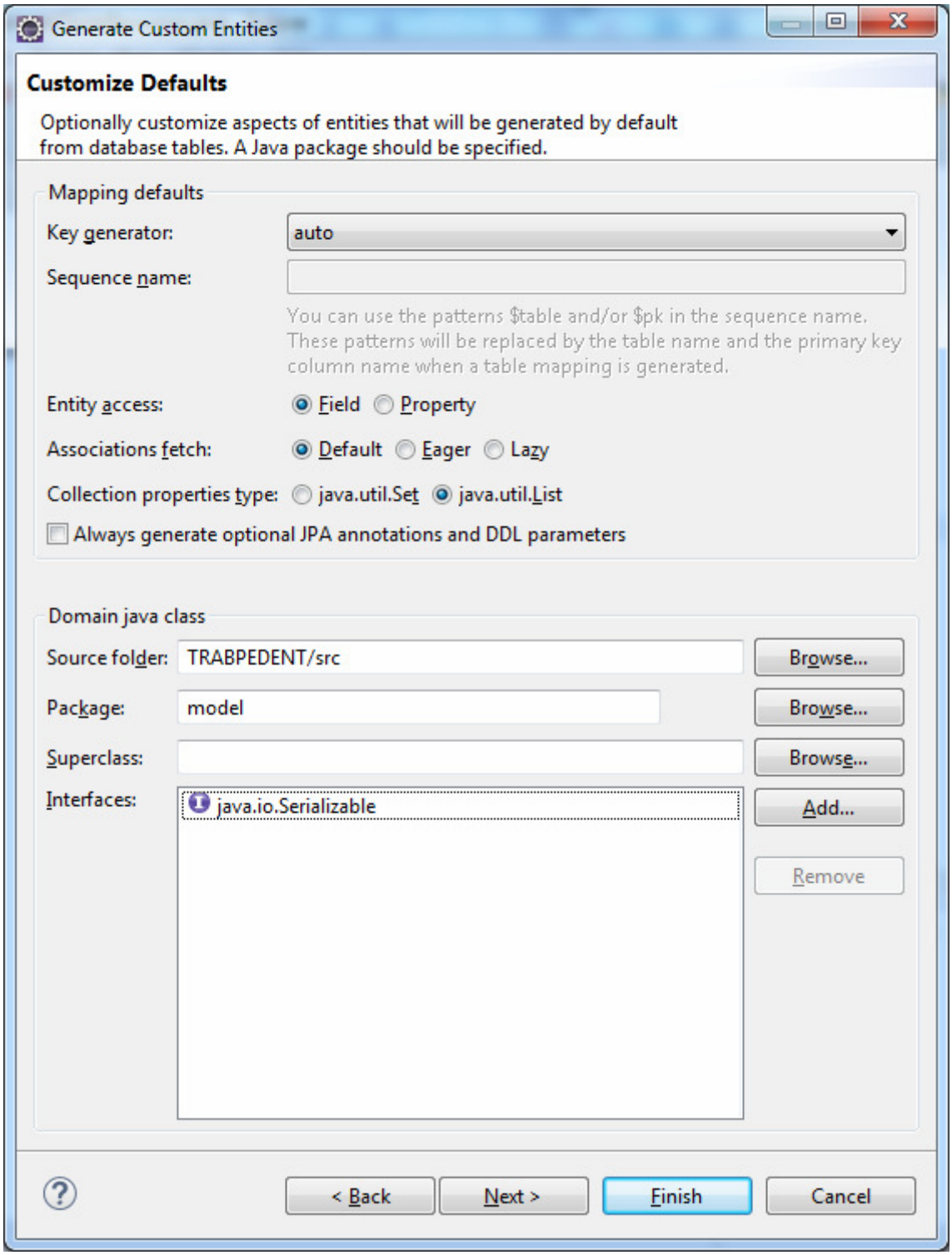
Selecione a conexão que você criou, o nome do esquema (nome do banco de dados que você criou para o trabalho, que deve ser *trabped*) e depois selecione as tabelas cujas entities serão geradas.

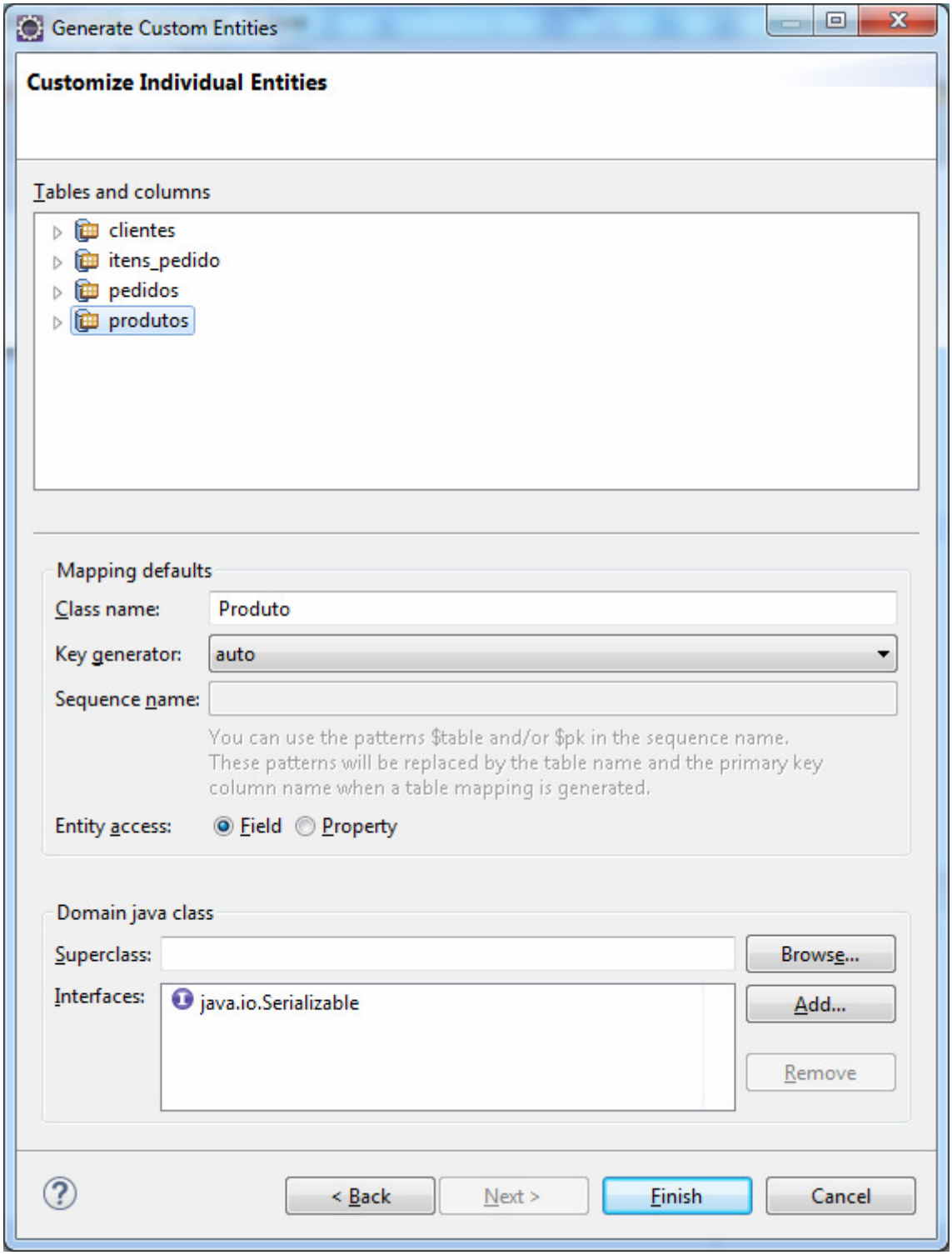
7) O wizard cria todas as relacionamentos entre as tabelas, convenientemente:



8-Continue pressionando Next na sequência de telas. Atente para as opções de geração:

Entity access: Como Campo (Field) ou propriedade (Property)
Associations Fetch (como os dados são retornados): Eager (todos de uma vez, mais rápido o acesso mas pode consumir muita memória) Lazy (por demanda, mais lento porém consome menos memória).
Collection properties type: em relações 1:n, o lado n pode ser gerado como *set* ou *list.*


Generate Custom Entities
Customize Defaults
Optionally customize aspects of entities that will be generated by default from database tables. A Java package should be specified.
Mapping defaults
Key generator:
auto
Sequence name:
You can use the patterns $table and/or $pk in the sequence name. These patterns will be replaced by the table name and the primary key column name when a table mapping is generated.
Entity access:
Field
Property
Associations fetch:
Default
Eager
Lazy
Collection properties type:
java.util.Set
java.util.List
Always generate optional JPA annotations and DDL parameters
Domain java class
Source folder:
TRABPEDENT/src
Browse...
Package:
model
Browse...
Superclass:
Browse...
Interfaces:
java.io.Serializable
Add...
Remove
< Back
Next >
Finish
Cancel



9-Abrir o arquivo persistence.xml (você vai encontrá-lo dentro na árvore do projeto, dentro de JPA Content) e incluir as seguintes linhas dentro de <persistence-unit> antes de </persistence-unit>

```
<properties>
        <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/trabped"/>
        <property name="javax.persistence.jdbc.user" value="root"/>
```

```
        <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver"/>
        <property name="javax.persistence.jdbc.password" value=""/>
         <property name="javax.persistence.target-database"
value="database"/>

 </properties>
```

10-Crie uma classe de nome TesteEntidades e "cole" todo o código abaixo nela. Em seguida execute essa classe clicando com o botão direito sobre o nome da mesma, em seguida *Run as* e depois *Java Application.*

```
package model;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Set;
import java.util.Iterator;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;



public class TesteEntidades {
        private EntityManagerFactory emf;
        private EntityManager em;
        private String PERSISTENCE_UNIT_NAME = "TRABPEDENT";

        /**
         * @param args
         */
        public static void main(String[] args) {

                TesteEntidades hello = new TesteEntidades();
            hello.initEntityManager();
            hello.read();
            hello.closeEntityManager();


        }
        private void read() {
            Pedido p =      (Pedido)em.createQuery("Select p from Pedido p where
p.codped=1").getSingleResult();


            List<ItensPedido> ip = p.getItensPedidos();
            Iterator itip = ip.iterator();
            while (itip.hasNext()){
              ItensPedido item = (ItensPedido)itip.next();
```

```
            Produto produto = item.getProduto();
            Pedido pedido = item.getPedido();
            Cliente cliente = p.getCliente();
            System.out.println ("Pedido: "+pedido.getCodped() );
            System.out.println("Data: "+pedido.getDatped());
            System.out.println("Cliente: "+cliente.getNomcli());
            System.out.println("Produto: "  + produto.getNompro());
            System.out.println("Quantidade: " + item.getQtditem());
            System.out.println("Valor unitario: "+ item.getValoritem());
              }
        }

        private void initEntityManager() {
            emf =
Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
            em = emf.createEntityManager();
        }

        private void closeEntityManager() {
            em.close();
            emf.close();
        }



}
```