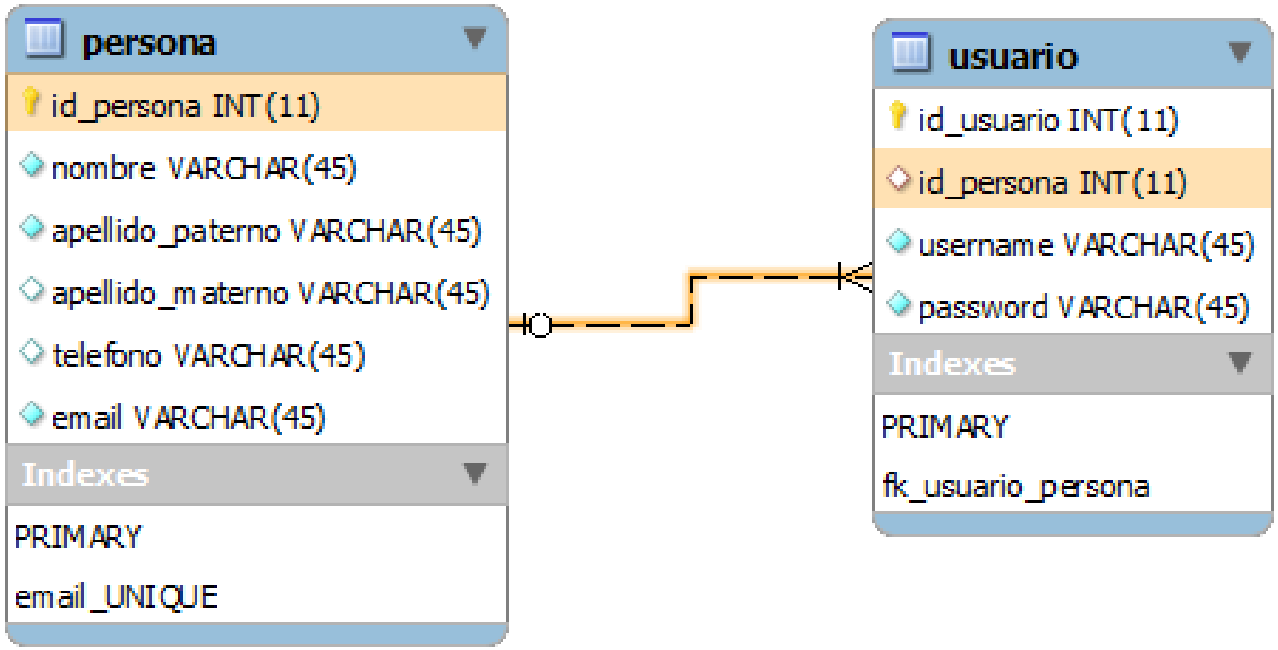
Global Mentoring

Ejercicio 8

Ingeniería Inversa con JPA y Eclipse IDE

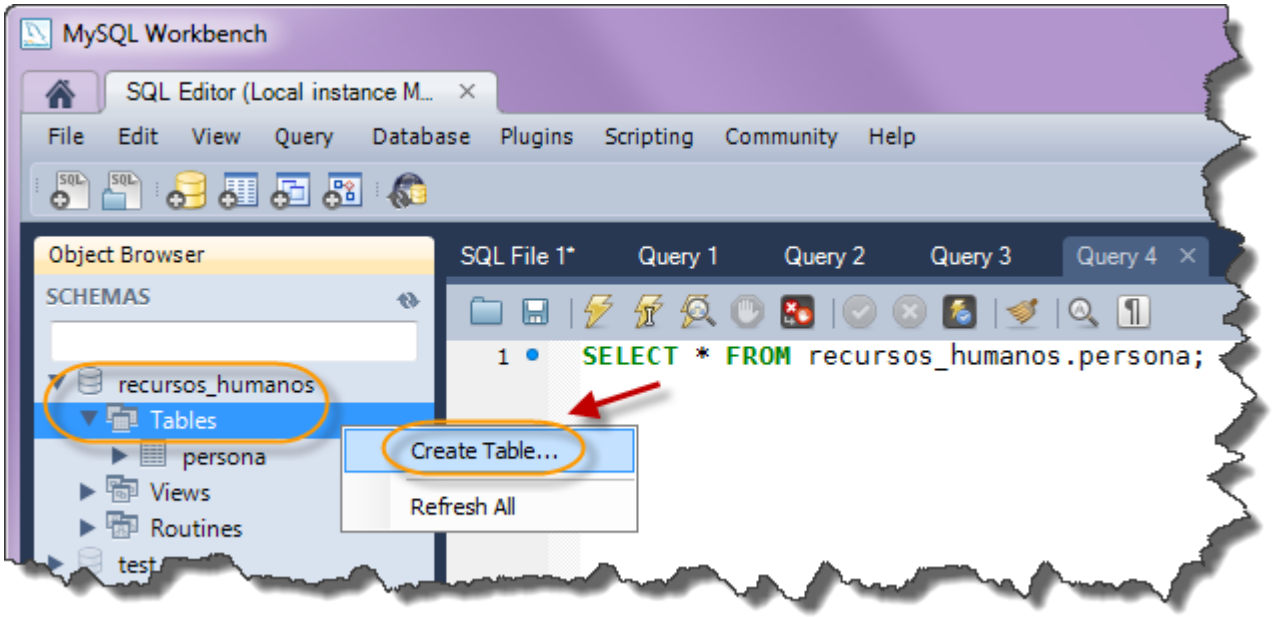
Objetivo del Ejercicio

- El objetivo del ejercicio crear el código de la tabla de Usuario y su relación con la tabla de Persona por medio del proceso de Ingeniería Inversa utilizando JPA y Eclipse IDE. El esquema entidad-relación queda como sigue:



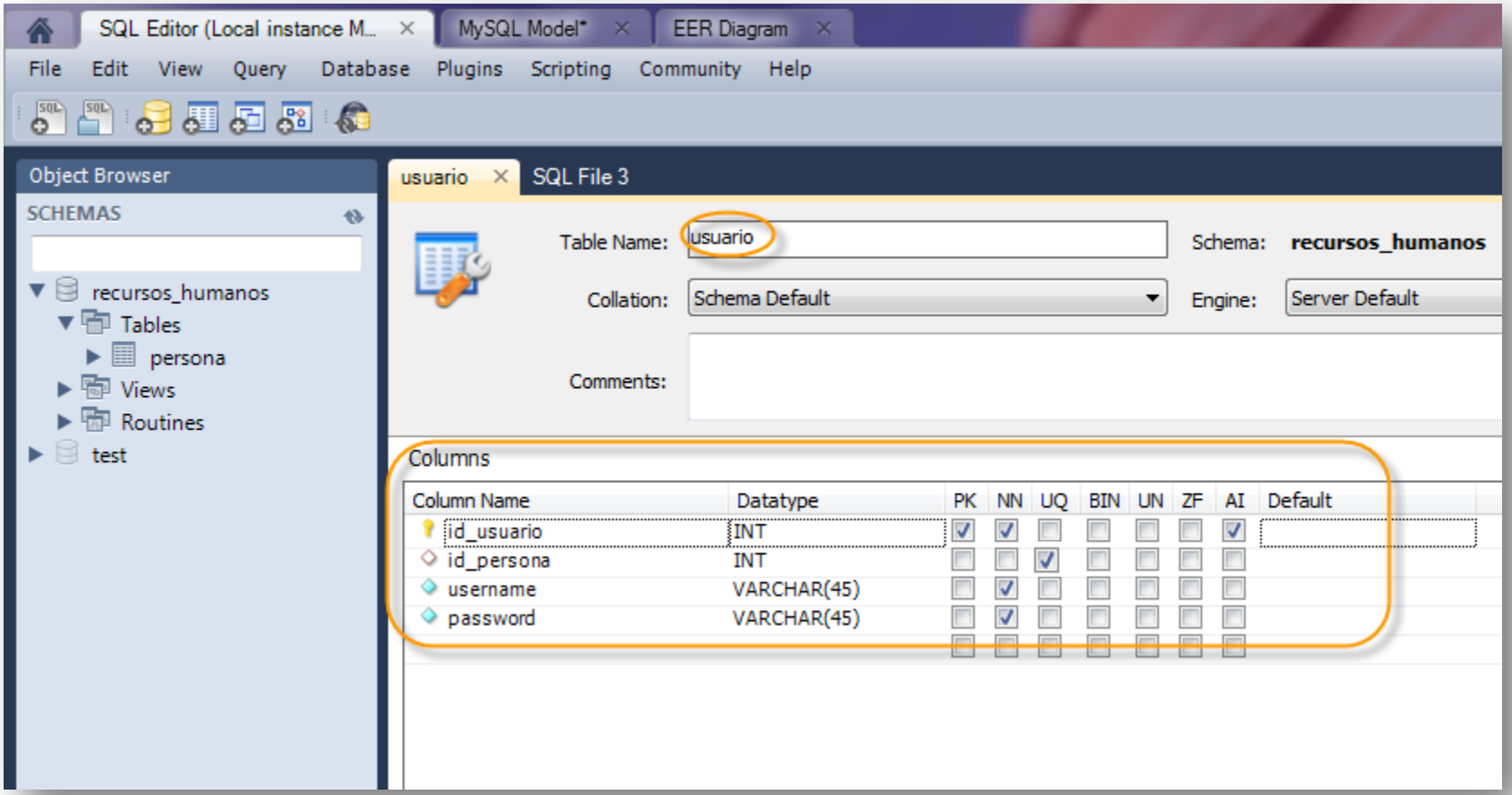
Paso 1. Crear la tabla Usuarios

Abrir MySql Workbench y agregar la tabla de Usuarios:



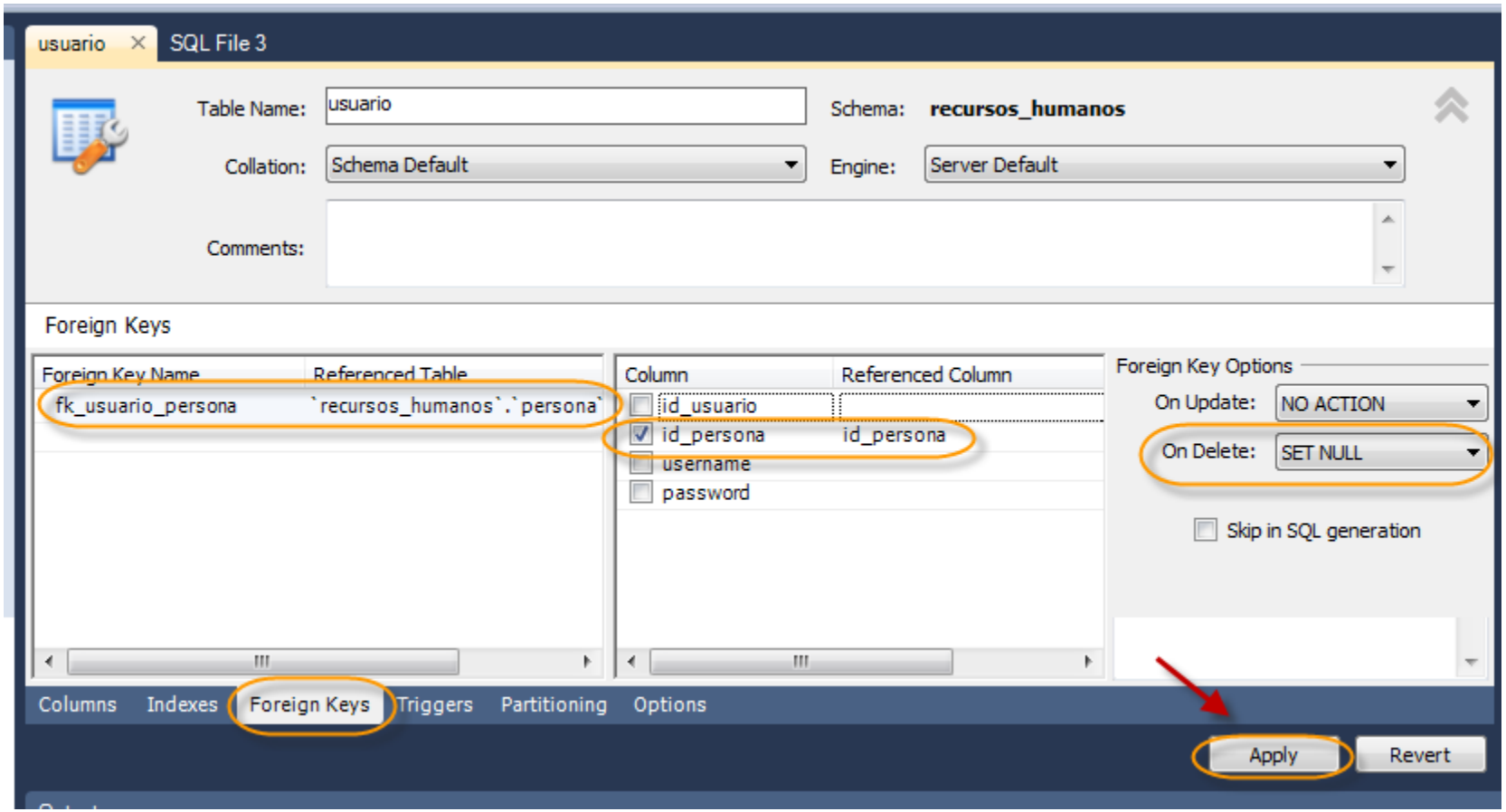
Paso 1. Crear la tabla Usuarios (cont)

Agregamos los siguientes campos a la tabla de Usuarios:



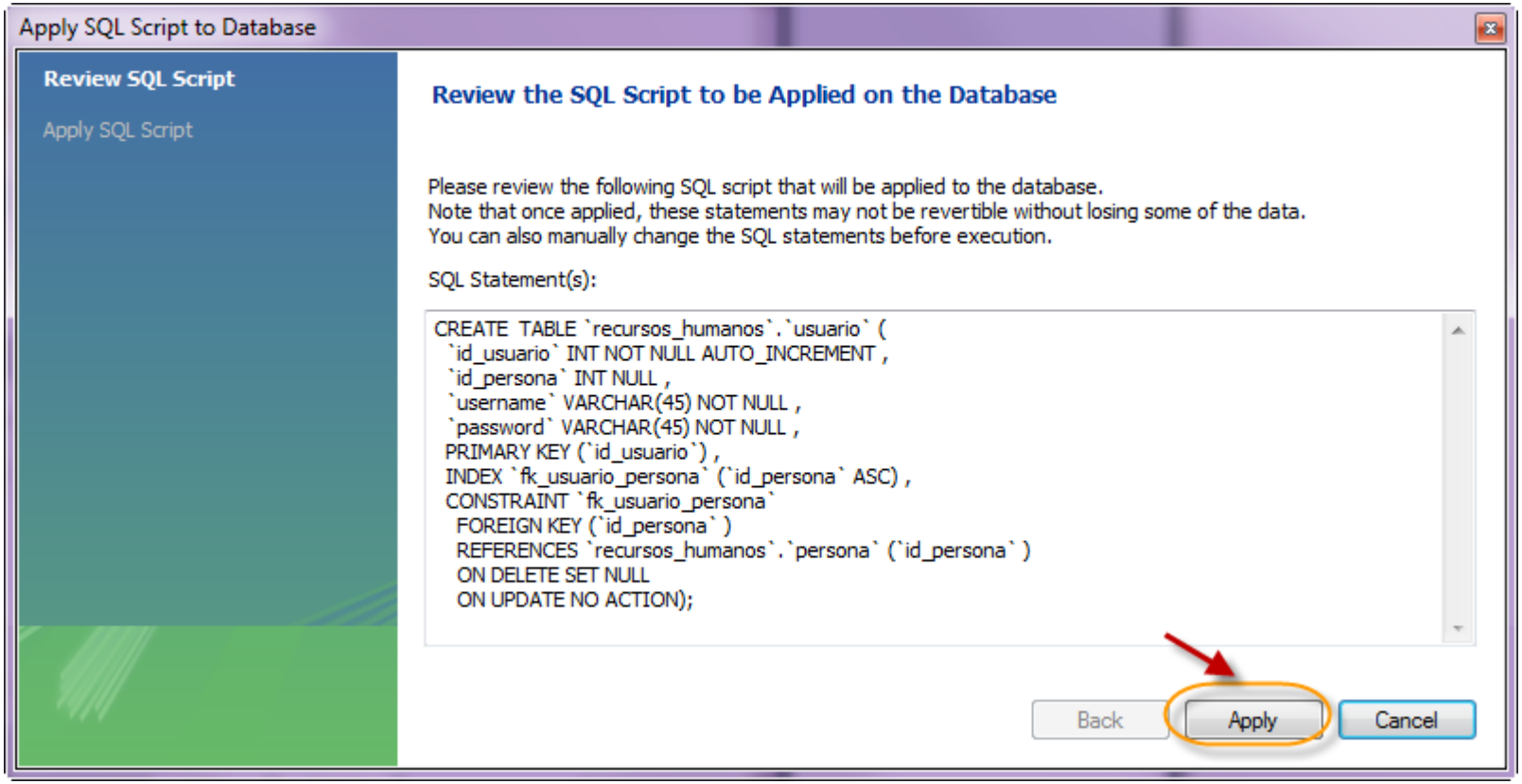
Paso 1. Crear la tabla Usuarios (cont)

Agregamos la configuración de la relación entre las tablas creando la llave foránea respectiva:



Paso 1. Crear la tabla Usuarios (cont)

Agregamos los siguientes campos a la tabla de Usuarios:



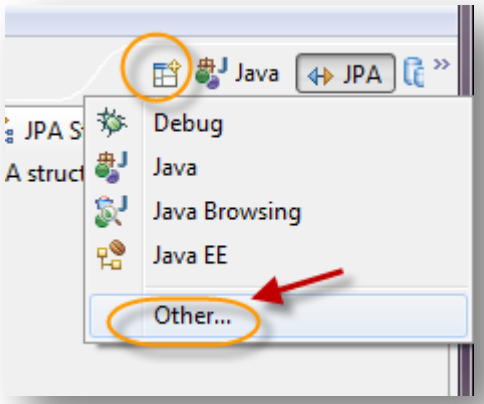
Paso 1. Crear la tabla Usuarios (cont)

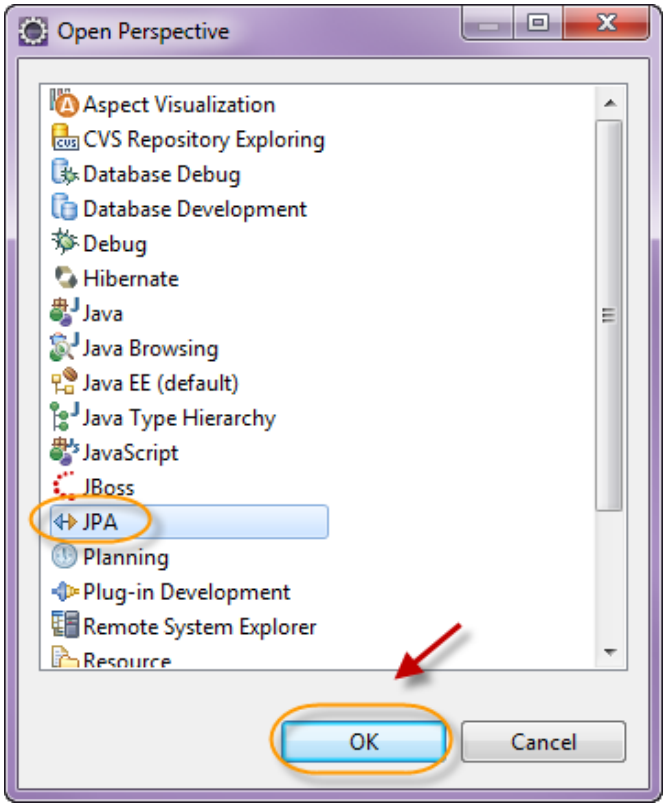
Insertamos un valor de prueba (el id_persona debe ser un id válido):

```
INSERT INTO usuario (id_persona, username, password) VALUES (1, 'jperez', '123');
```

Paso 2. Ingeniería Inversa con JPA

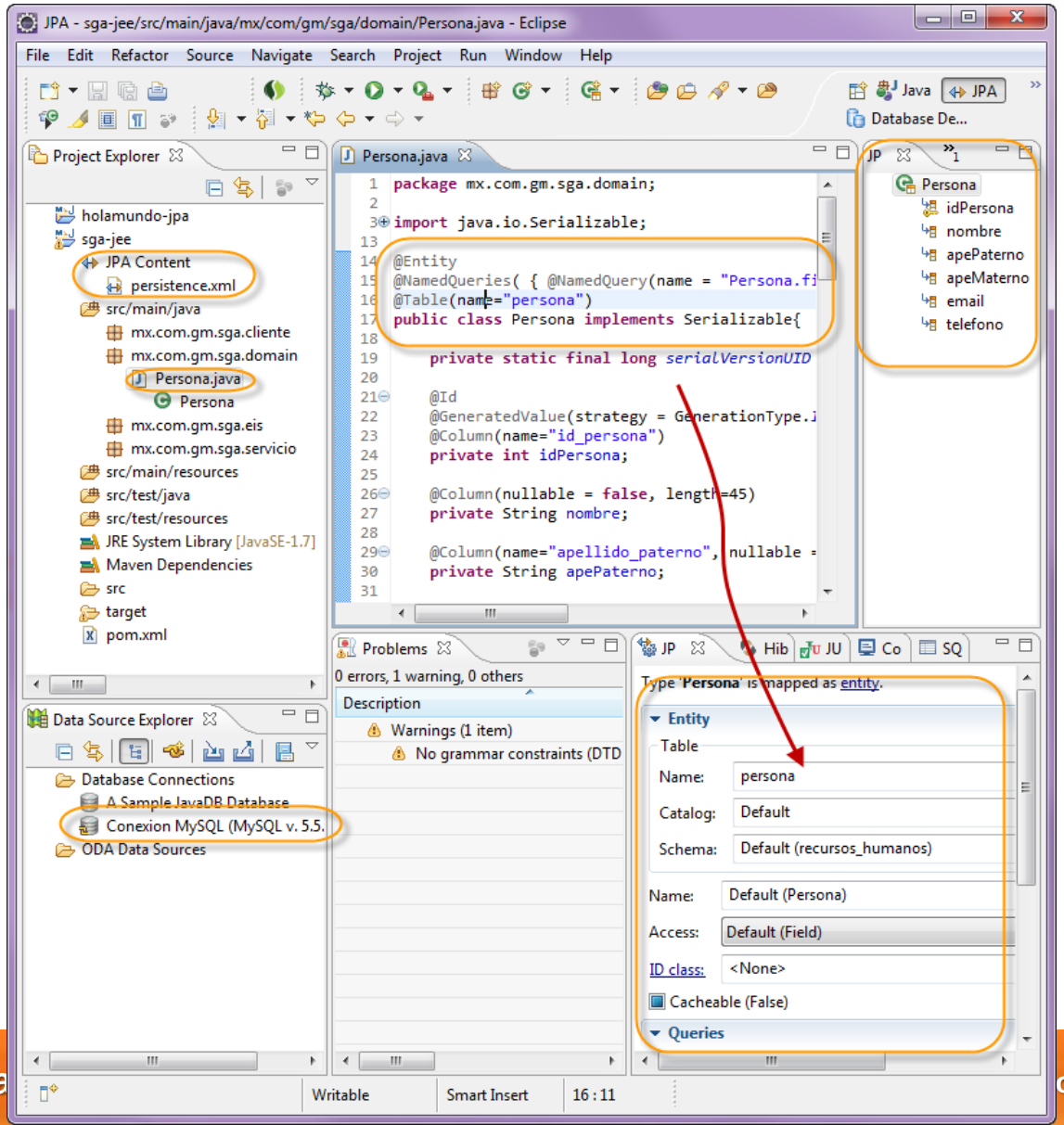
Con la configuración JPA de nuestro proyecto, y la configuración de la conexión MySql del IDE, vamos a revisar el proceso de Ingeniería Inversa, para generar el código Java a partir de la Base de Datos. Cambiamos a la vista Java EE:





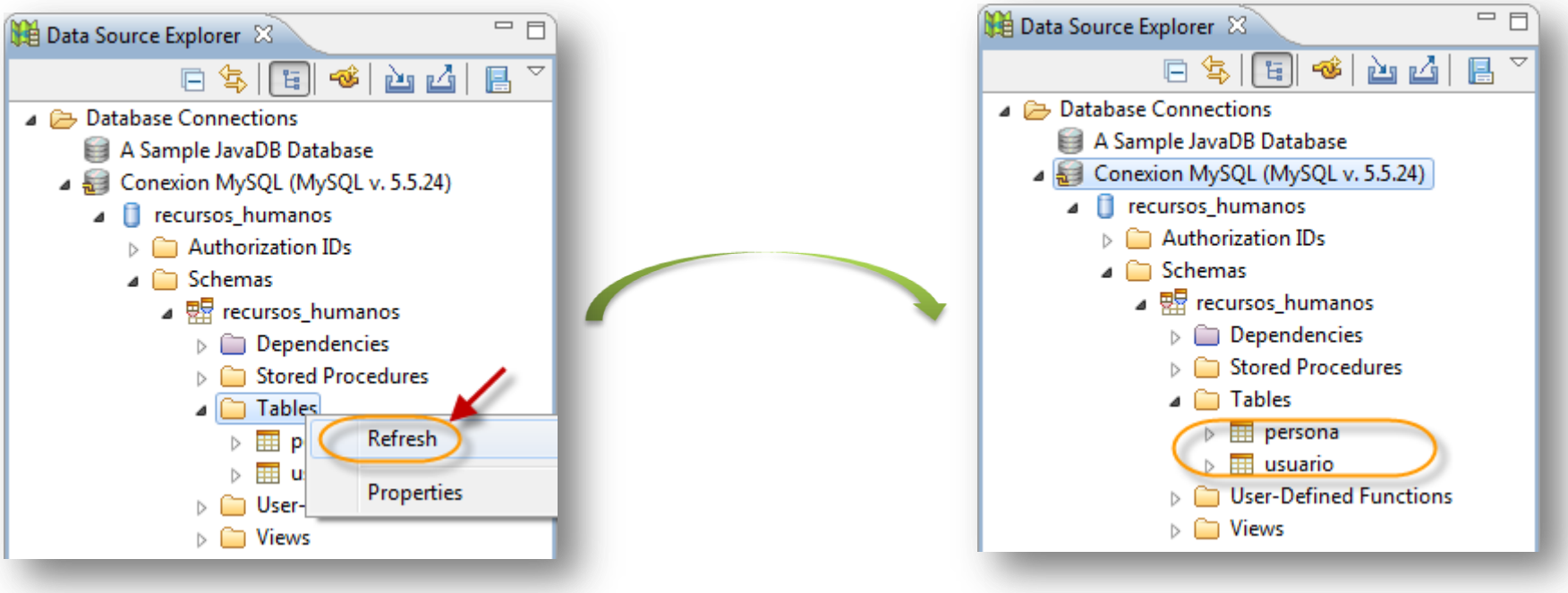
Paso 2. Ingeniería Inversa con JPA (cont)

Vista JPA:



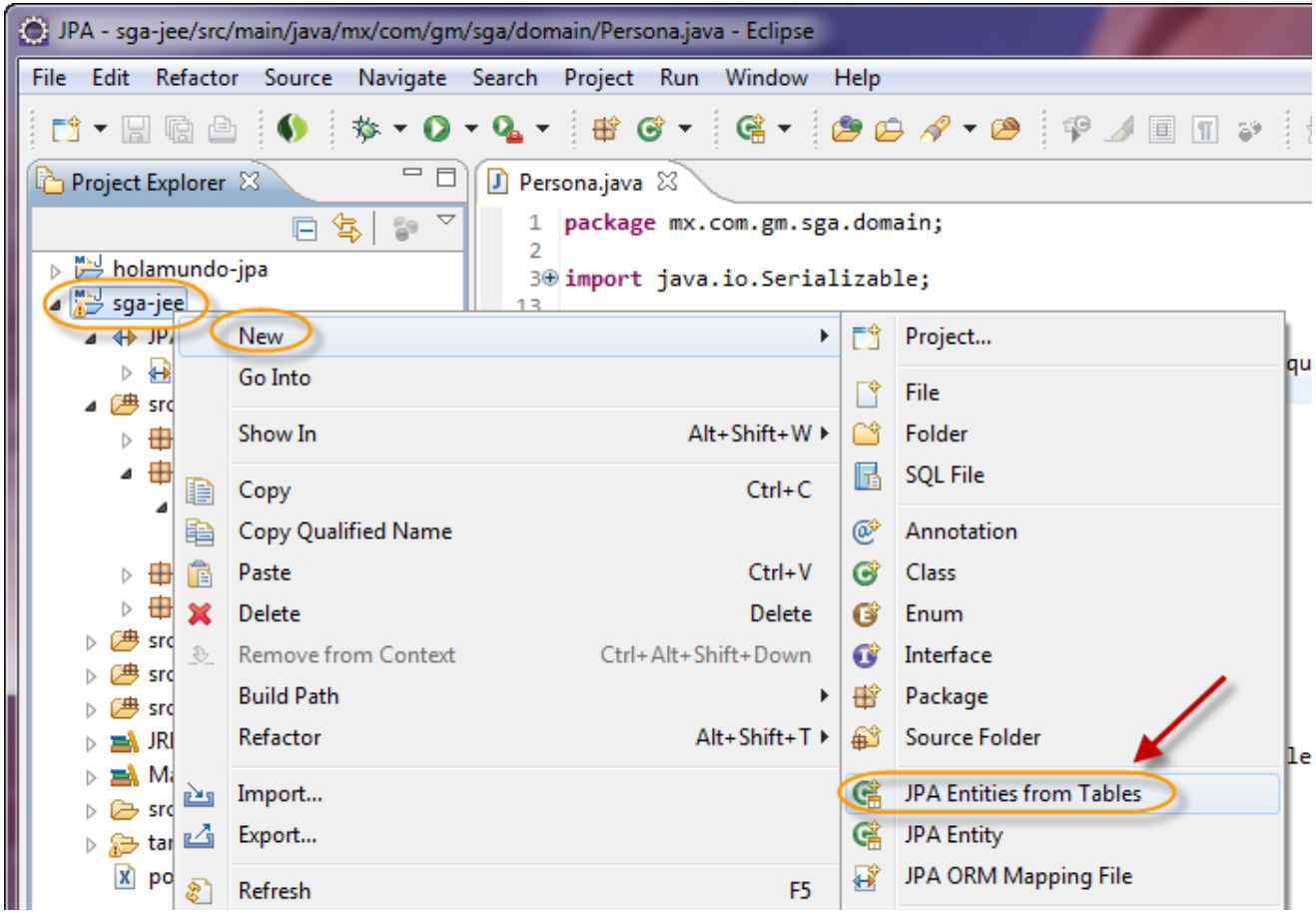
Paso 2. Ingeniería Inversa con JPA (cont)

Hacemos un refresh para ver las 2 tablas:



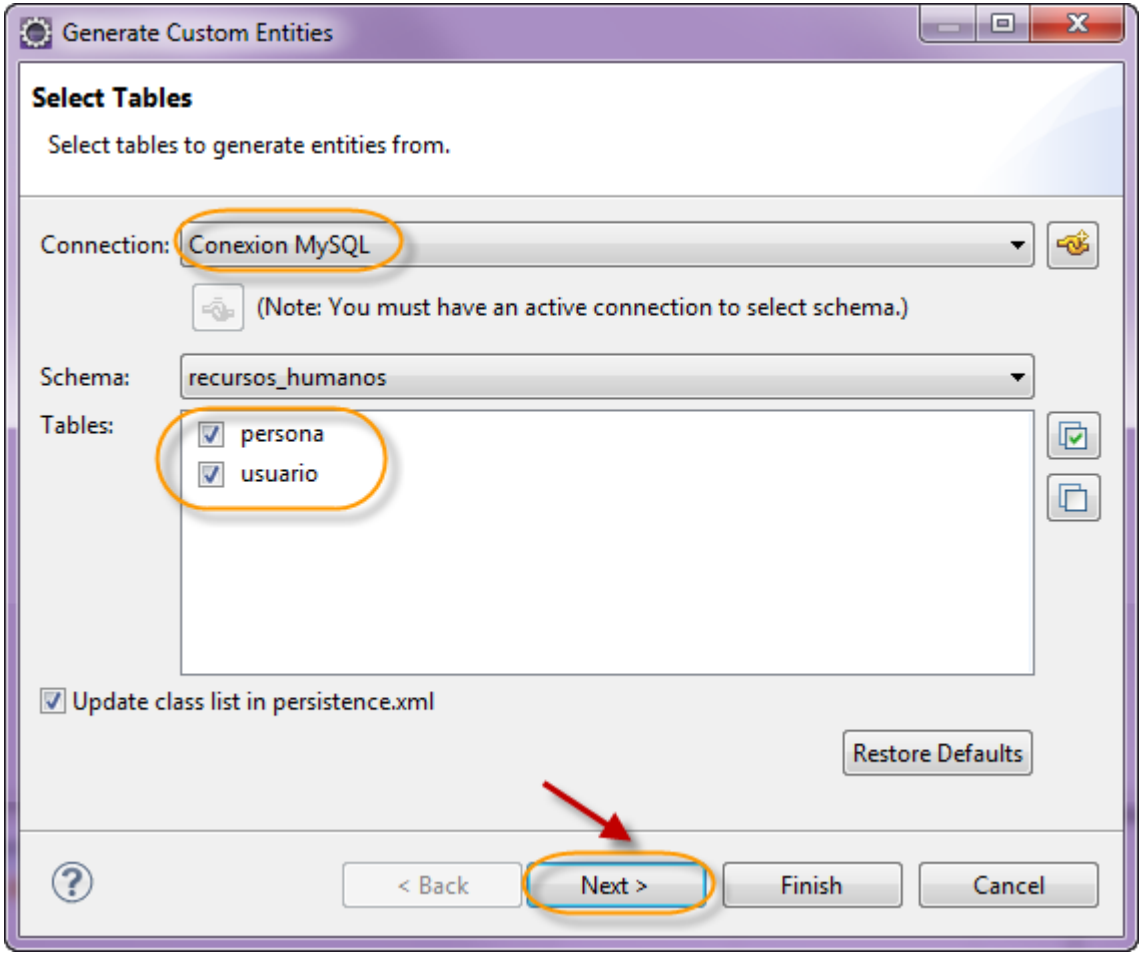
Paso 2. Ingeniería Inversa con JPA (cont)

Utilizamos el Wizard de JPA Entities from Tables:



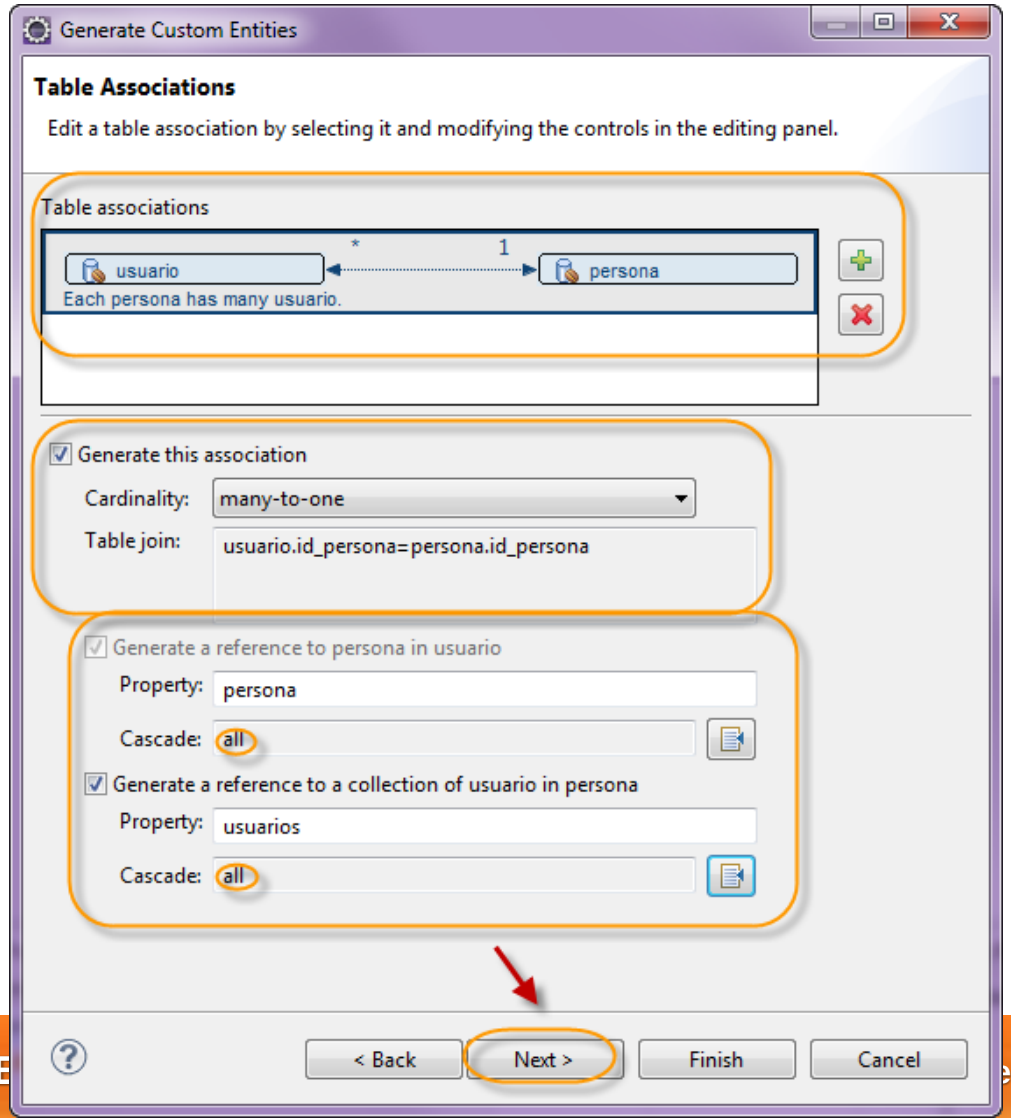
Paso 2. Ingeniería Inversa con JPA (cont)

Utilizamos el Wizard de JPA Entities from Tables:



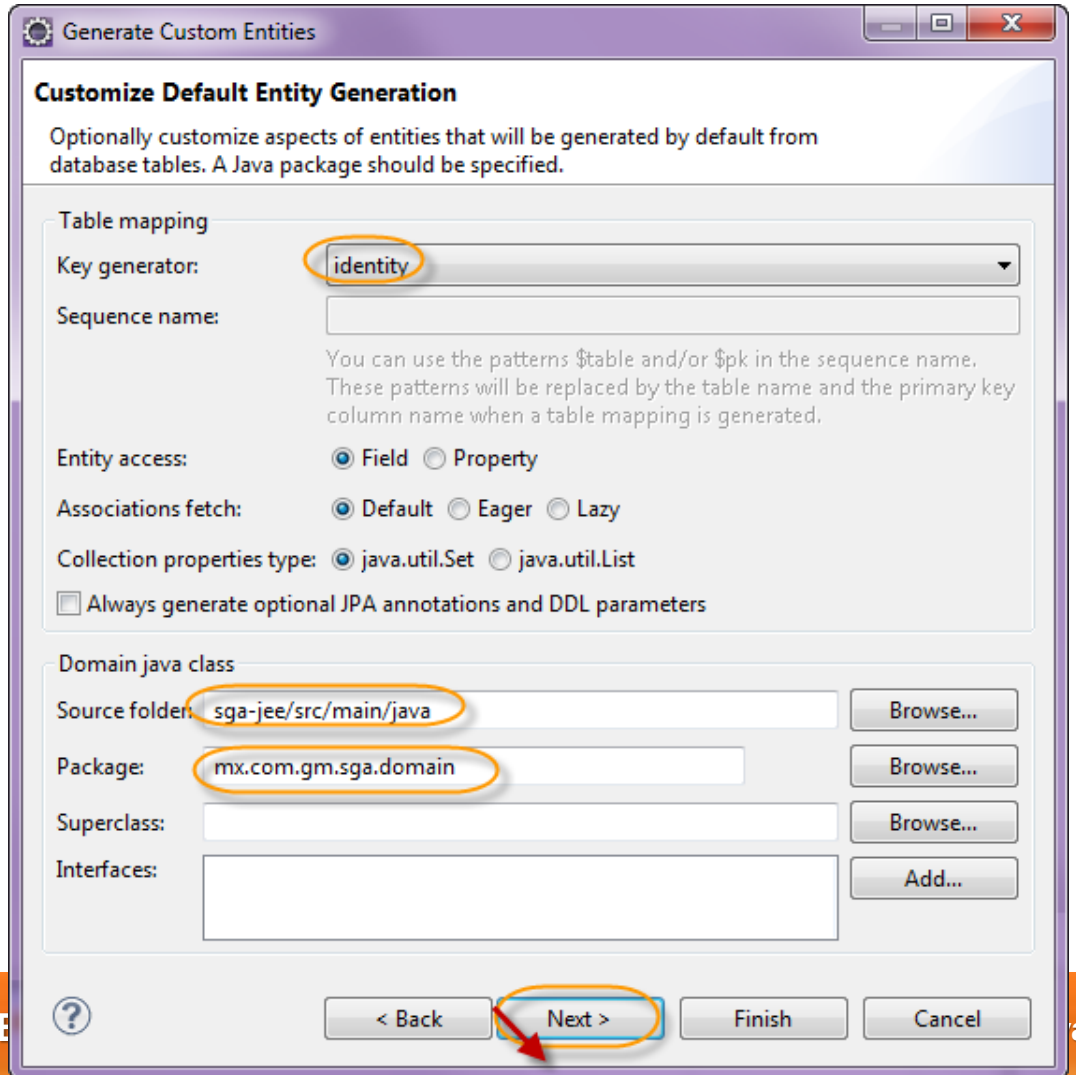
Paso 2. Ingeniería Inversa con JPA (cont)

Utilizamos el Wizard de JPA Entities from Tables:



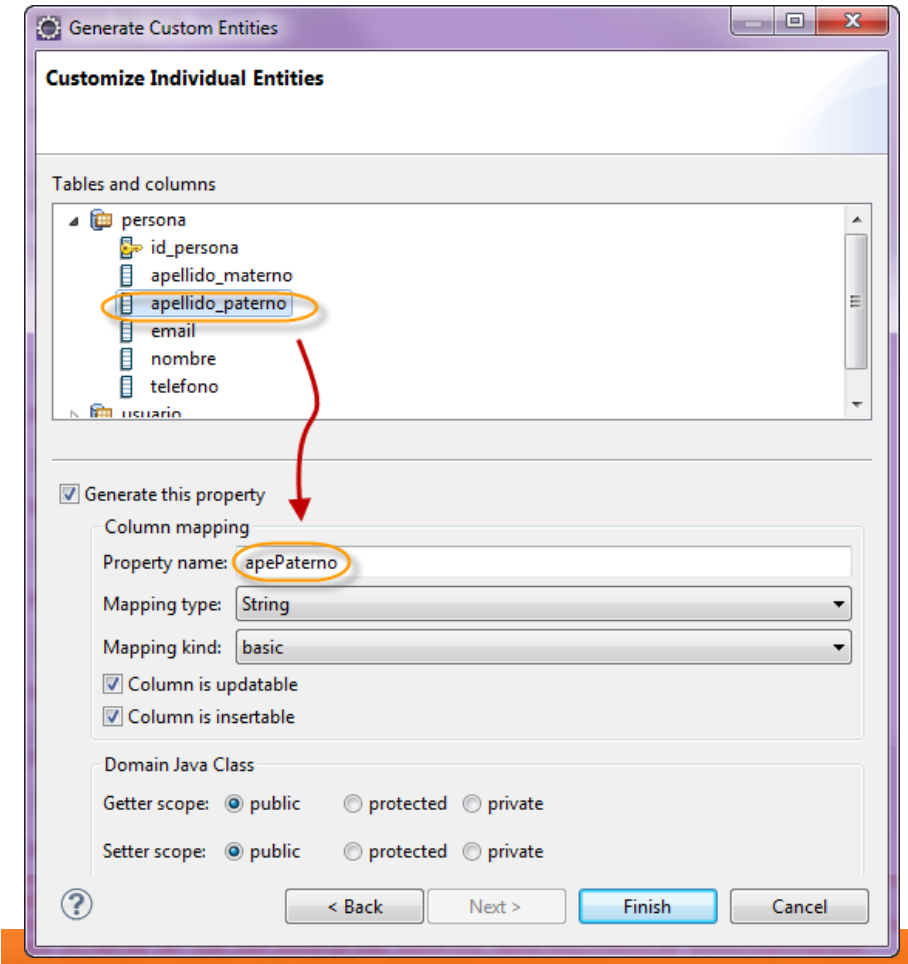
Paso 2. Ingeniería Inversa con JPA (cont)

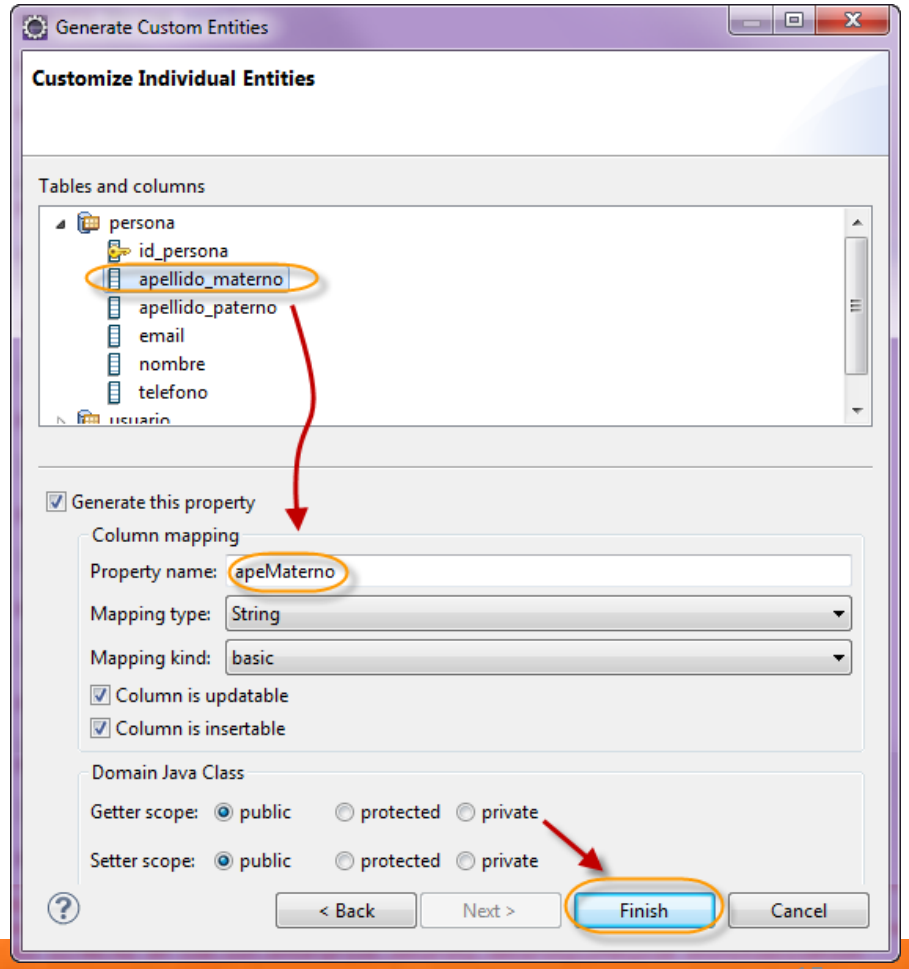
Utilizamos el Wizard de JPA Entities from Tables:



Paso 2. Ingeniería Inversa con JPA (cont)

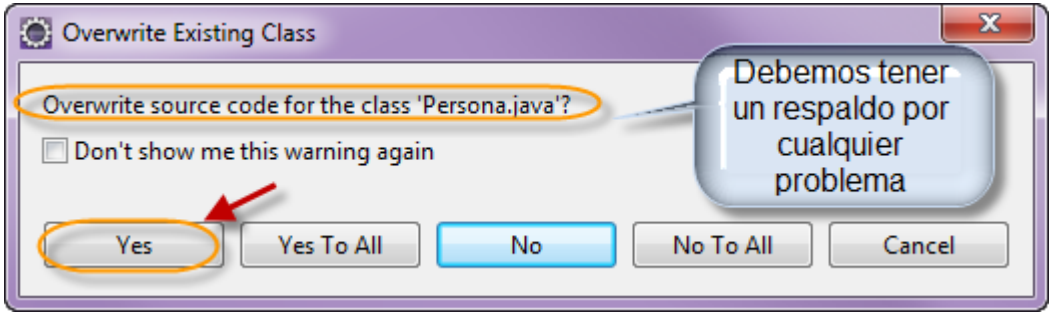
Modificamos los valores por default de los atributos apePaterno y apeMaterno para tener el mismo nombre que habíamos manejado:





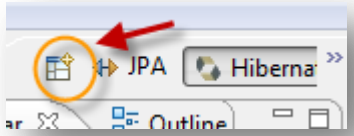
Paso 2. Ingeniería Inversa con JPA (cont)

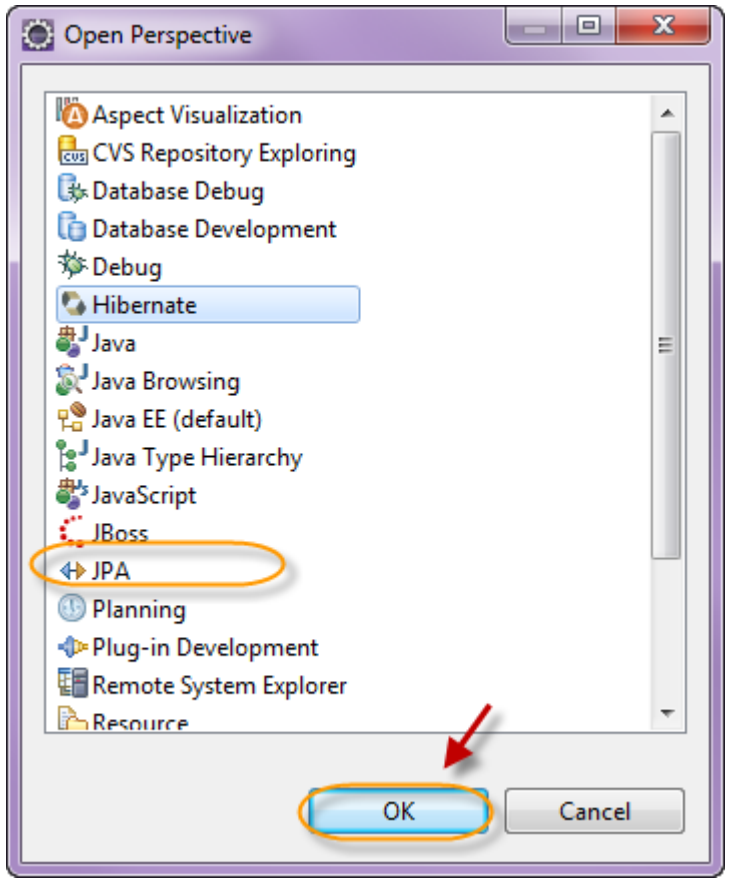
Debido a que ya habíamos creado manualmente la clase de entidad Persona, va a sobreescribir la clase. Debemos tener un respaldo del código por cualquier detalle en el proceso de Ingeniería Inversa:



Paso 3. Complementar el Código JPA

Cambiamos a la vista JPA:





Paso 3. Complementar el Código JPA (cont)

Agregamos los constructores que tenía la clase Persona:

```
public Persona(int idPersona) {
      this.idPersona = idPersona;
  }

  public Persona(int idPersona, String nombre, String apePaterno, String apeMaterno,
        String email, String telefono) {
      this.idPersona = idPersona;
      this.nombre = nombre;
      this.apePaterno = apePaterno;
      this.apeMaterno = apeMaterno;
      this.email = email;
      this.telefono = telefono;
  }

  public Persona(String nombre, String apePaterno, String apeMaterno,
        String email, String telefono) {
      this.nombre = nombre;
      this.apePaterno = apePaterno;
      this.apeMaterno = apeMaterno;
      this.email = email;
      this.telefono = telefono;
  }
```

Paso 3. Complementar el Código JPA (cont)

Agregamos los constructores a la clase Usuario:

```
public Usuario(int idUsuario, String password, String username,
        Persona persona) {
    this.idUsuario = idUsuario;
    this.password = password;
    this.username = username;
    this.persona = persona;
}

public Usuario(String username, String password,
        Persona persona) {
    this.username = username;
    this.password = password;
    this.persona = persona;
}

public Usuario(String username, String password) {
    this.username = username;
    this.password = password;
}
```

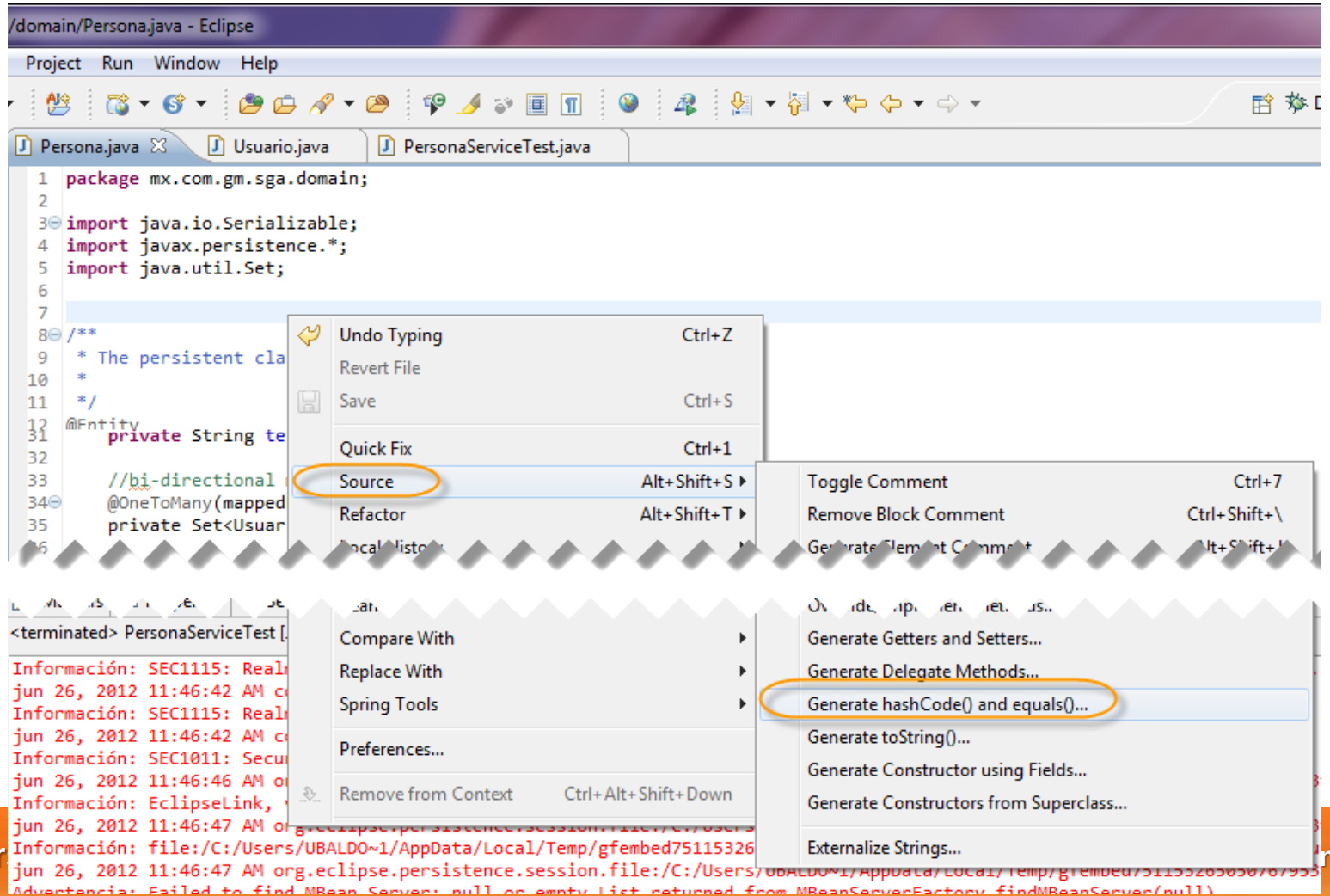
Paso 3. Complementar el Código JPA (cont)

Agregamos los métodos siguientes a nuestras clases de entidad Persona y Usuario respectivamente:

- toString()
- equals()
- hashCode()

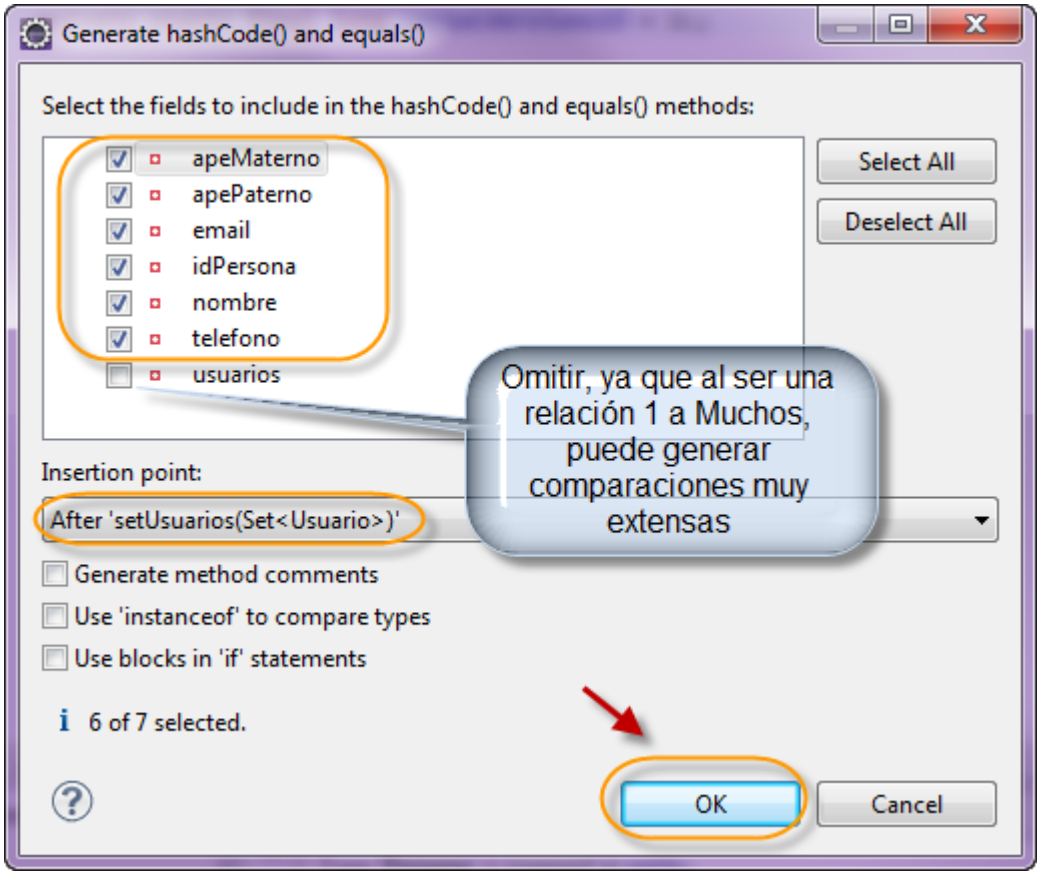
Paso 3. Complementar el Código JPA (cont)

Agregamos los métodos equals y hashCode a la clase Persona:



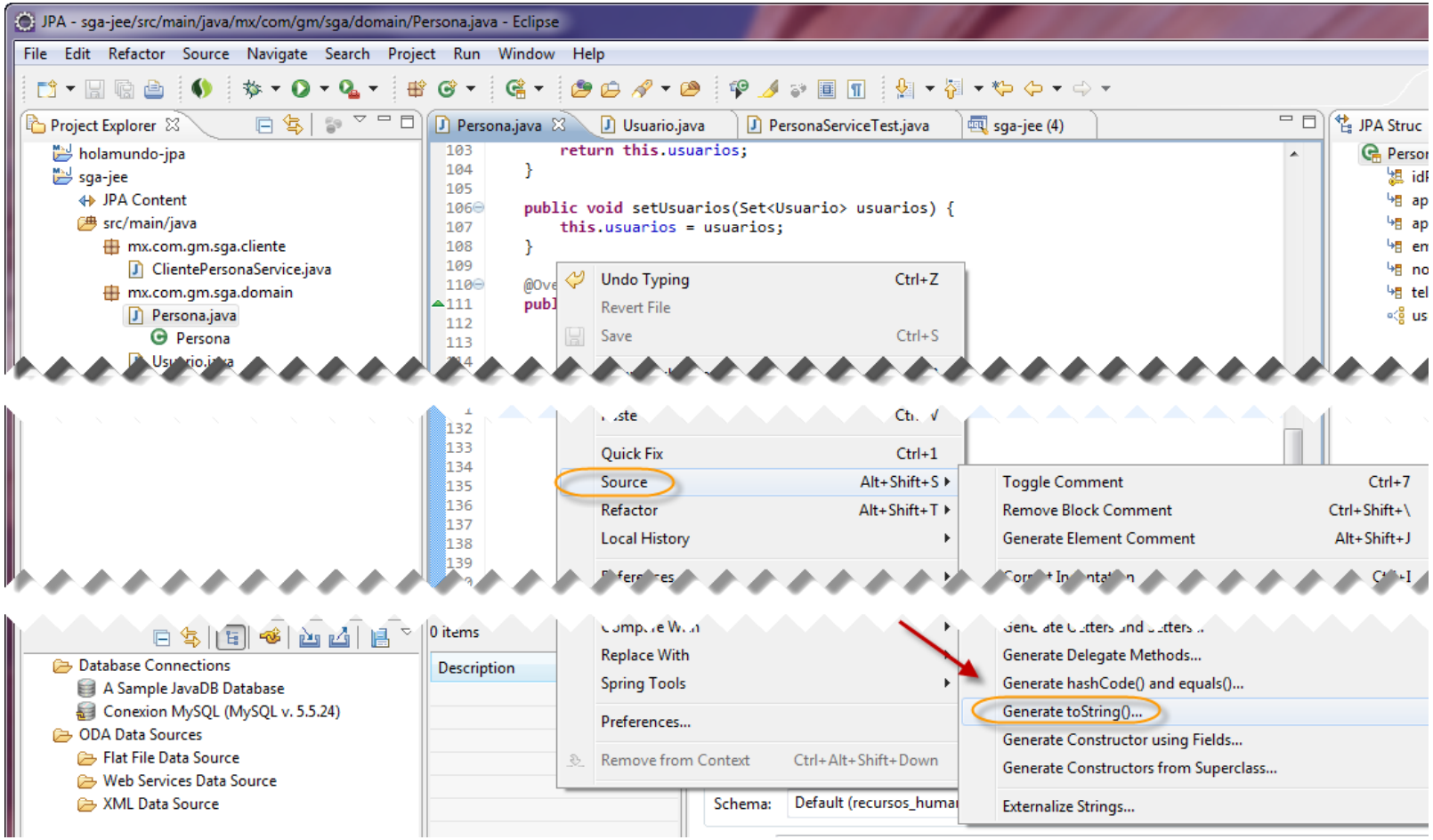
Paso 3. Complementar el Código JPA (cont)

Agregamos los métodos equals y hashCode a la clase Persona:



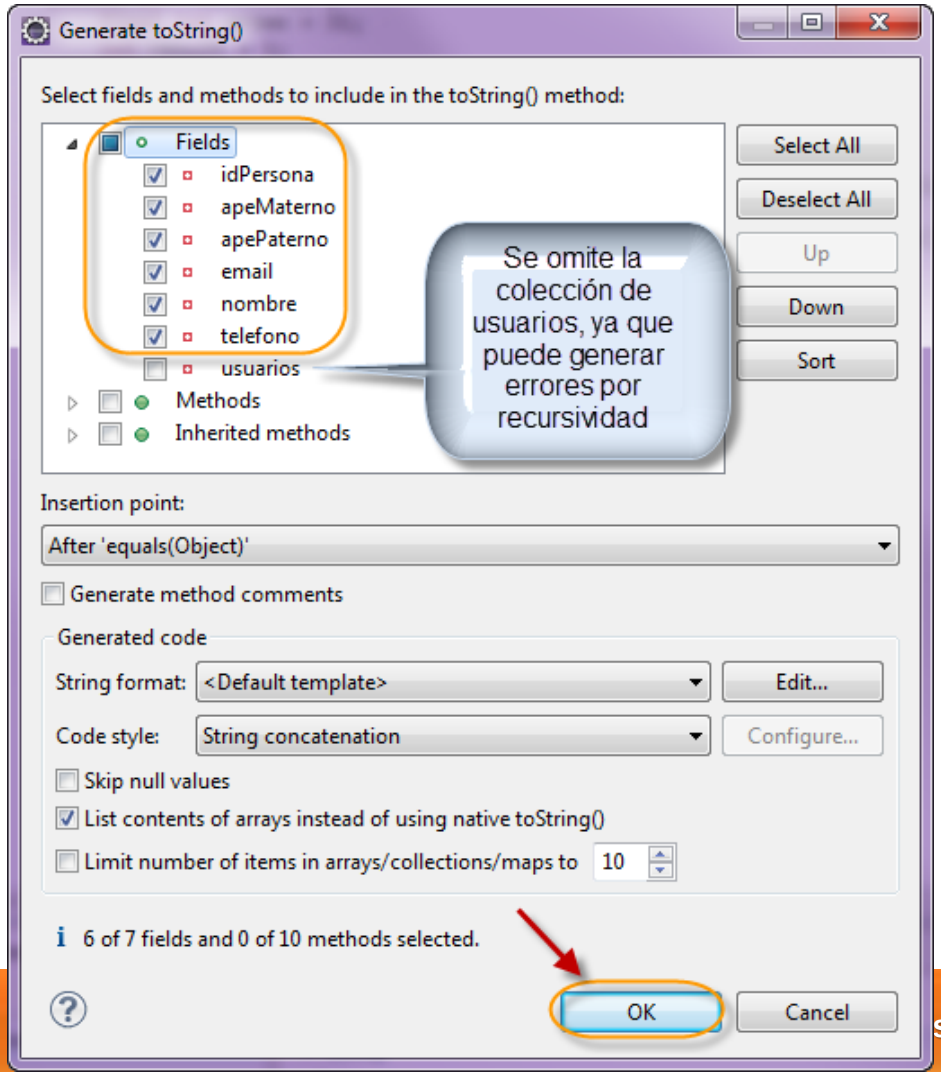
Paso 3. Complementar el Código JPA (cont)

Agregamos el método toString() a la clase Persona:



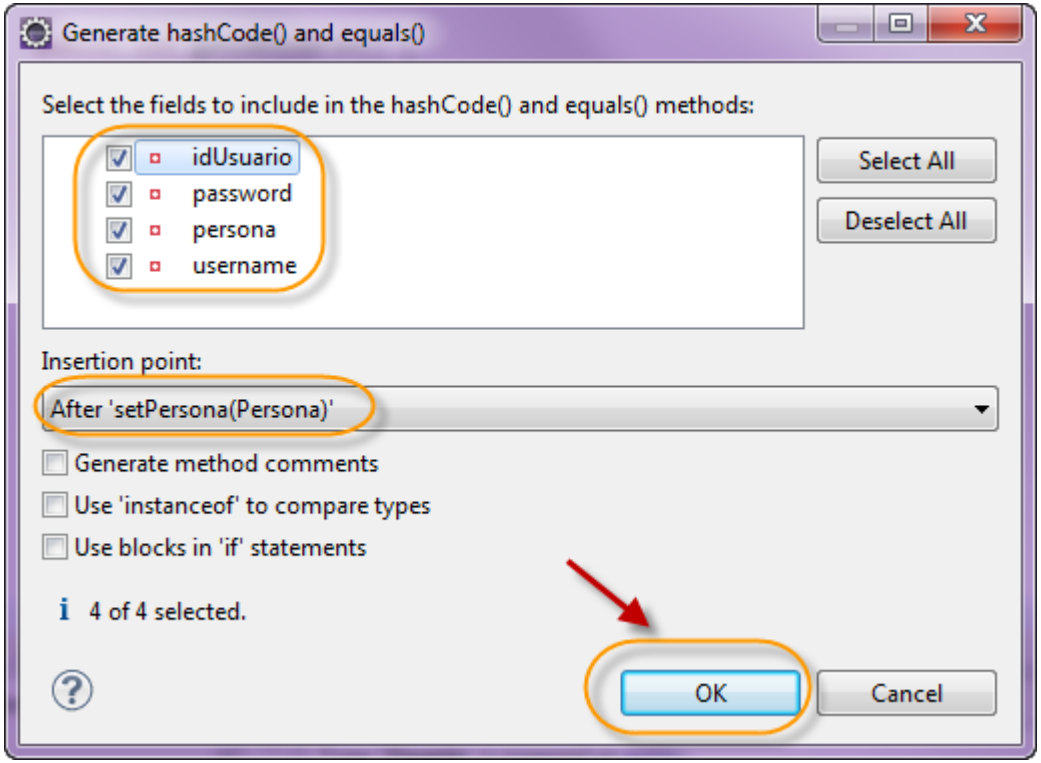
Paso 3. Complementar el Código JPA (cont)

Agregamos el método toString() a la clase Persona:



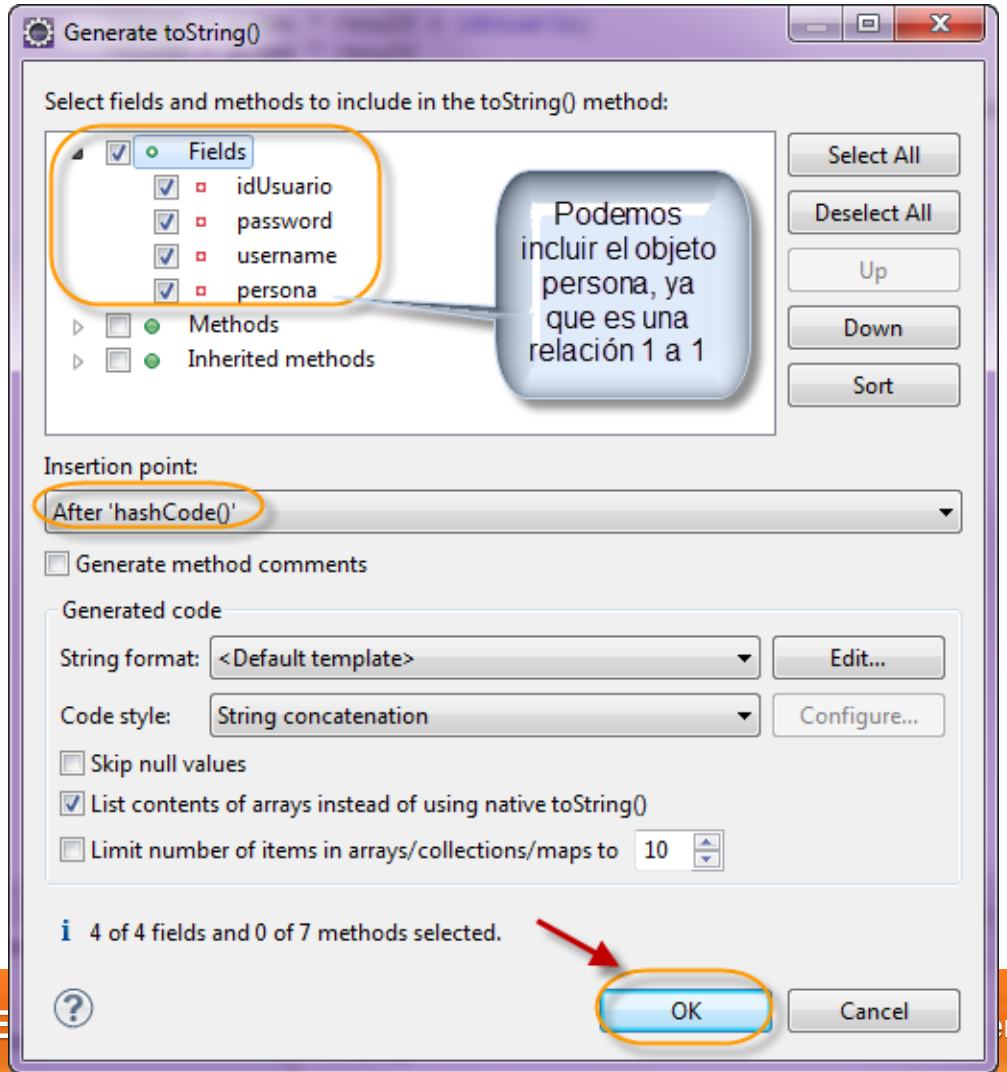
Paso 3. Complementar el Código JPA (cont)

Agregamos los métodos equals y hashCode a la clase Usuario:



Paso 3. Complementar el Código JPA (cont)

Agregamos el método toString() a la clase Persona:



Paso 3. Complementar el Código JPA (cont)

Agregamos los queries en las clases de Entidad, los cuales deben ir después de la anotación @Entity de cada clase.

- Clase Persona:

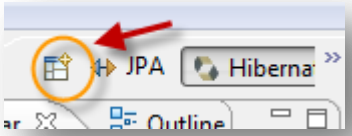
```
@NamedQueries( { @NamedQuery(name = "Persona.findAll", query = "SELECT p FROM Persona p ORDER BY p.idPersona") })
```

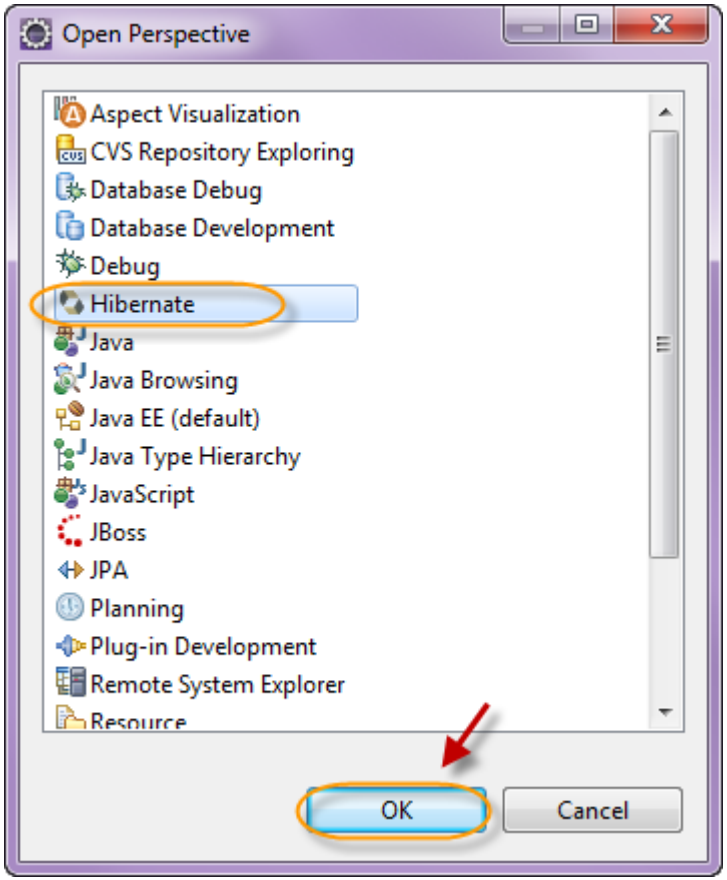
- Clase Usuario:

```
@NamedQueries( { @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM Usuario u ORDER BY u.idUsuario") })
```

Paso 4. Comprobación vía consola HQL Editor

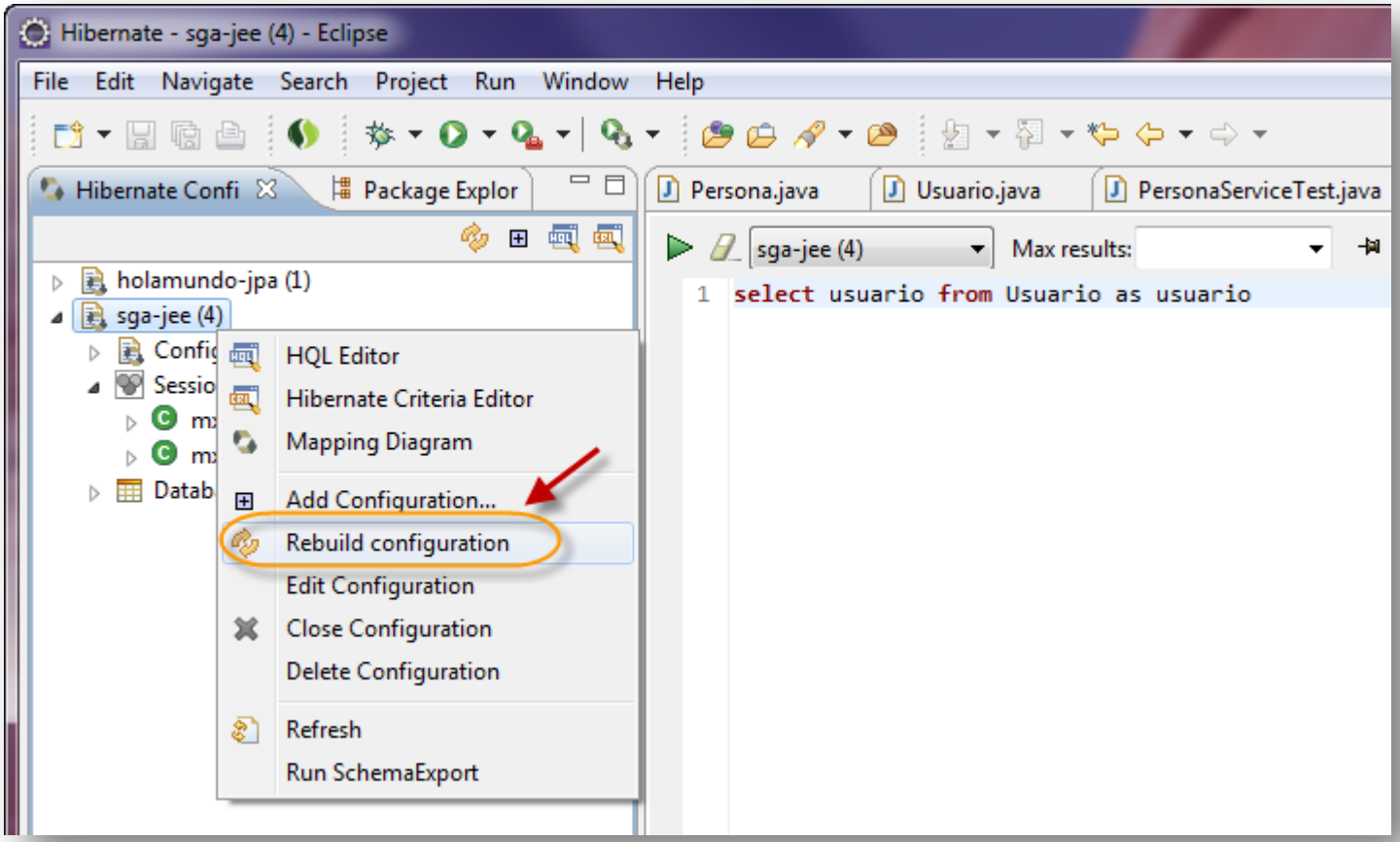
Nos cambiamos a la vista de Hibernate y recargamos el código Java, para que reconozca los cambios la consola de Hibernate.





Paso 4. Comprobación vía consola HQL Editor

Reconstruimos la configuración de la sesión de Hibernate para que reconozca los cambios realizados en nuestro código Java:



Paso 4. Comprobación vía consola HQL Editor

Comprobamos que visualicemos el registro que obtiene el objeto Usuario:

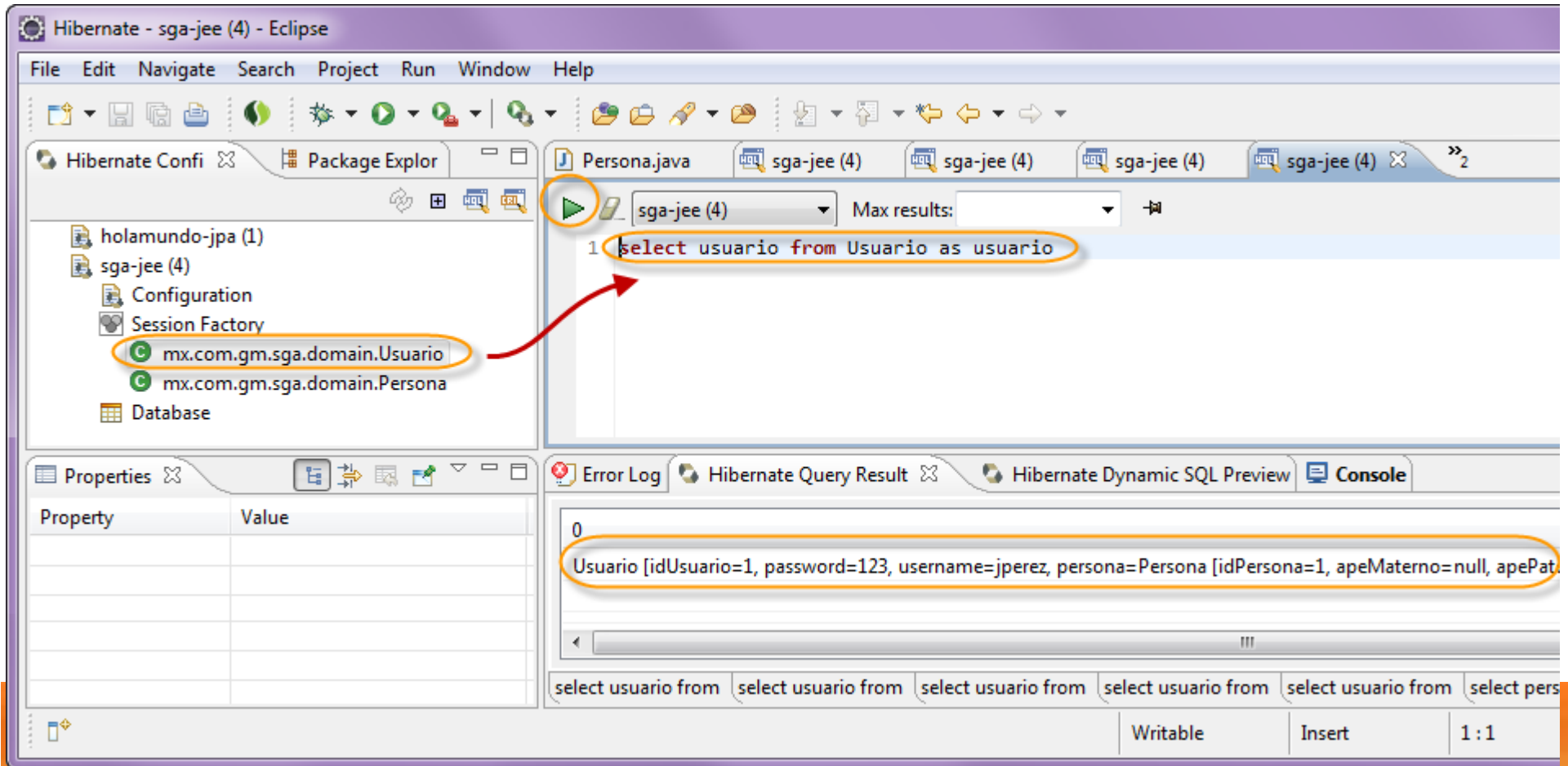
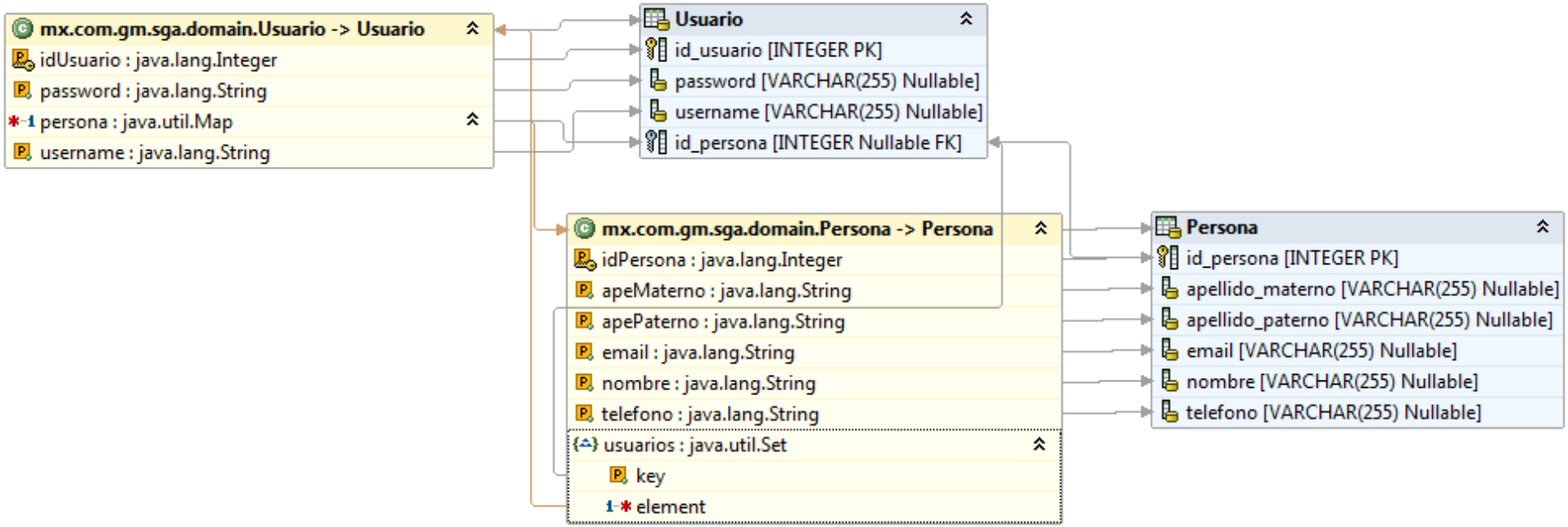


Diagrama de Mapeo Final

La ingeniería inversa que realizamos tiene como conclusión el siguiente diagrama de Mapeo, donde podemos observar las clases Java *(en amarillo)* y las tablas de base de datos *(en azul)*:



Ejercicio y Conclusión

Se deja como ejercicio crear las clases de la capa de datos (*UsuarioDao*, *UsuarioDaoImpl*), la capa de servicio (*UsuarioService y UsuarioServiceImpl*), así como la prueba unitaria (*UsuarioServiceTest*), y comprobar que se puedan realizar consultas sobre esta clase de Entidad desde la capa de Servicio. Solamente se solicita crear la interfaz local, no es necesario crear una interfaz remota.

Conclusión:

Con este ejercicio pudimos observar cómo podemos generar código Java a partir de un esquema de base de datos ya creado, esto nos permitirá agilizar el código JPA respectivo y nos permitirá enfocarnos en el negocio de nuestro proyecto.

Si no contamos con un esquema Entidad-Relación, JPA permite crear las tablas de base de datos asociadas a las clases de Entidad, esto es posible debido al mapeo entre el código Java y el SQL que genera JPA para la creación de las tablas de la base de datos. Esto se configura en el archivo persistence.xml, sin embargo frecuentemente nos encontraremos con esquemas entidad-relación ya creados, por lo que esta técnica puede ahorrarnos mucho tiempo en la generación de código Java y JPA.

www.globalmentoring.com.mx

Pasión por la tecnología Java

Experiencia y Conocimiento para tu vida