CUESTIONES Y EJERCICIOS

1. Conecta con la base de datos COMPANY y muestra en pantalla los registros de la tabla REGIONS.

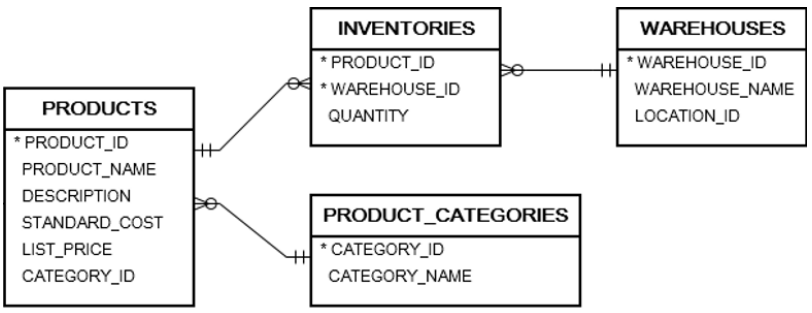
2. Conecta con la base de datos COMPANY y muestra en pantalla los registros de la tabla COUNTRIES.

3. Conecta con la base de datos COMPANY y muestra en pantalla el nombre de la región y el nombre del país ordenados por el nombre de región.

4. Crea una clase PruebaInsert que conecte con la base de datos COMPANY y amplíe el inventario de 'Mexico City' para que ahora disponga de 10 unidades de cada uno de los siguientes productos:

PRODUCT_ID	PRODUCT_NAME	DESCRIPTION
7	G.Skill Ripjaws V Series	Speed:DDR4-3200,Type:288-pin DIMM,CAS:15Module:8x8GBSize:64GB
8	Intel Xeon E5-1650 V4	Speed:3.6GHz,Cores:6,TDP:140W
9	Intel Xeon E5-2640 V4	Speed:2.4GHz,Cores:10,TDP:90W

   Extracto del modelo de tablas del esquema:



5. Crea una clase PruebaUpdate para que el inventario de 'Mexico City' vea incrementado el stock de todas las CPU que comercializa en 10 unidades.

6. Crea una clase PruebaDelete que conecte con la base de datos COMPANY para que el almacén de 'Mexico City' deje de comercializar cualquier tipo de CPU eliminando estos registros de la tabla de inventario.

7. Crea una clase PruebaInsertPrepared que le pida por teclado al usuario las características de una nueva CPU y se inserte el nuevo producto en la tabla PRODUCTS, utilizando un PreparedStatement para ello.

8. Crea una clase PruebaSelectPrepared que le pida por teclado al usuario la categoría de los productos que desea consultar, así como el precio mínimo y máximo de los mismos. Además, se le pide al usuario si quiere que los resultados se muestren ordenados por precio ascendente o descendente. La clase debe utilizar PreparedStatement para realizar la consulta con los parámetros definidos por el usuario y ofrecer los resultados.

9. Tomando como punto de partida el ejercicio 7 vamos a hacer una transacción que permita dar de alta simultáneamente un nuevo producto pero que pertenezca a una nueva categoría. La idea es pedirle al usuario los datos del producto y cuando llegue el momento de asignarlo a una categoría ofrecerle la opción de crear una nueva. Cuando hayamos recopilado todos los datos iniciamos una transacción en la que primero insertamos la nueva categoría, recuperamos su clave primaria (mediante ROWID o mediante una SELECT que busca por el nombre de la categoría recién insertada) y después insertamos el nuevo producto.

10. Vamos a hacer una nueva transacción que permita borrar un almacén (WAREHOUSE) y se borren, además, todas sus entradas en el inventario. Para ello debes pedirle al usuario que identifique el almacén a eliminar y que confirme la operación. Acto seguido debes comenzar una transacción que elimine en primer lugar los registros de inventario y después el registro del almacén.

11. Implementa la clase Cliente que se asociará a la tabla CUSTOMERS de la BD COMPANY tomando el siguiente diagrama de clase:

Cliente
- propiedades asociadas a la tabla CUSTOMERS de la base de datos COMPANY
+ constructor con todas las propiedades + getters de todas las propiedades + setters de todas las propiedades excepto de customerId + toString() + void insert() throws AccesoClienteException + void update() throws ClienteNoExisteException, AccesoClienteException + void delete() throws ClienteNoExisteException, AccesoClienteException + static List<Cliente> selectAll() throws AccesoClienteException + static Cliente selectById (long customerId) throws ClienteNoExisteException, AccesoClienteException + static List<Cliente> selectByName(String name) throws AccesoClienteException

Teniendo en cuenta que:

- Se lanzará una AccesoClienteException si se produce alguna SQLException.
- Se lanzará una ClienteNoExisteException si no se encuentra el cliente buscado.
- Los métodos insert, update y delete realizan la acción correspondiente tomando los

valores de la query de las propiedades del objeto y tomando la propiedad *customerId* como clave primaria de la tabla.

NOTA: para las pruebas de este ejercicio se recomienda utilizar valores de *customerId* que sean >= 1000 para que no se mezclen con los ya existentes en la tabla.

- Los métodos selectAll, selectById y selectByName son estáticos, así que podrán ser utilizados directamente desde la referencia a la clase. Estos métodos deben buscar los Cliente según el criterio de búsqueda correspondiente. En el caso de *selectByName* es importante destacar que, si hay más de un cliente cuyo nombre coincida con el nombre buscado, todos deben ser incluidos en la lista