

**Alumno:** Kevin Zamora Amela

**Asignatura:** Acceso a datos

## Tarea 3 para AD04.

### Enunciado


Una empresa dispone de una base de datos que contiene las tablas con la información necesarias para realizar la gestión de sus empleados. Las tablas son las siguientes:

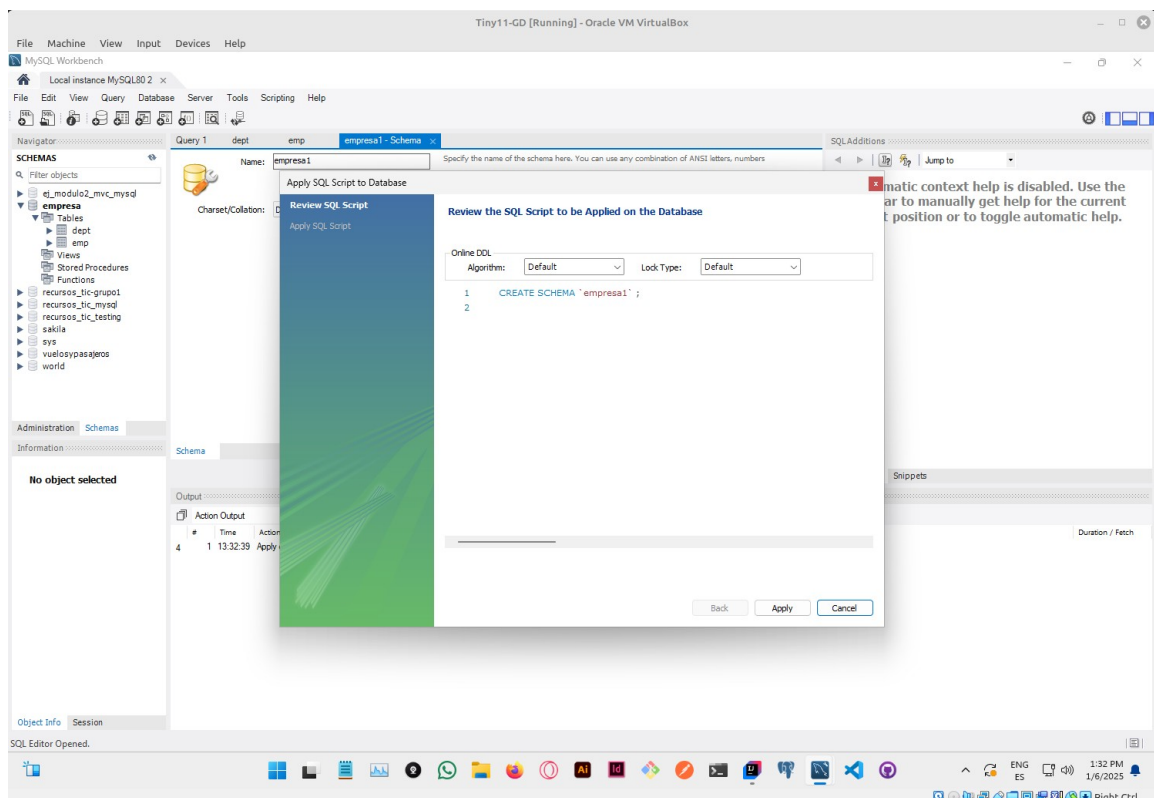
- Tabla DEPT que contiene información de los diferentes departamentos que tiene la empresa. La clave principal DEPTNO.
- Tabla EMP que contiene la información de los diferentes empleados que tiene la empresa. Tiene como clave principal EMPNO y como ajenas DEPTNO que relaciona con la tabla departamentos y MGR que establece la relación con la misma tabla (EMP) representando la relación ser jefe de.

Para la creación de las tablas y sus relaciones de partida, se adjuntaba un fichero o 'script' con las sentencias SQL necesarias, este se llamaba '**tablas\_mysql.sql**' que se adjunta.

En base al enunciado anterior, se han implementado los apartados siguientes:

#### 1. Crear la base de datos que se llamará 'empresa'.

- Accedemos a nuestro administrador de bases de datos, en nuestro caso se ha utilizado MySQL Workbench.
- Creamos una conexión a base de datos usando nuestro usuario ("user") y nuestra contraseña ("user4321") para acceder.
- Hacemos clic sobre la opción "Crear un nuevo esquema en el servidor conectado", la cual se encuentra representada por el siguiente icono . Una vez creada, ejecutamos el 'script' suministrado para llenarla con nuestros datos.



**Kevin Zamora**

## 2. Configura y crea la ORM Hibernate.

- Añadimos las dependencias necesarias, las cuales resultan las siguientes (en este caso):

```
m pom.xml (HibernateMysql) x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- object xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd" -->
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>com.kevinzamora</groupId>
6     <artifactId>HibernateMysql</artifactId>
7     <version>1.0-SNAPSHOT</version>
8     <packaging>jar</packaging>
9     <properties>
10         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
11         <maven.compiler.release>21</maven.compiler.release>
12         <exec.mainClass>com.kevinzamora.hibernatemysql.HibernateMysql</exec.mainClass>
13     </properties>
14     <dependencies>
15         <!-- https://mvnrepository.com/artifact/org.hibernate.orm/hibernate-core -->
16         <dependency>
17             <groupId>org.hibernate.orm</groupId>
18             <artifactId>hibernate-core</artifactId>
19             <version>6.6.2.Final</version>
20         </dependency>
21         <!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->
22         <dependency>
23             <groupId>com.mysql</groupId>
24             <artifactId>mysql-connector-j</artifactId>
25             <version>8.4.0</version>
26         </dependency>
27         <dependency>
28             <groupId>org.testng</groupId>
29             <artifactId>testng</artifactId>
30             <version>RELEASE</version>
31             <scope>compile</scope>
32         </dependency>
33         <dependency>
34             <groupId>org.projectlombok</groupId>
35             <artifactId>lombok</artifactId>
36             <version>1.18.36</version>
37             <type>jar</type>
38         </dependency>
39     </dependencies>
40 </project>
```

- Creamos un archivo 'hibernate.cfg.xml' para alojar la configuración básica para habilitar nuestra conexión a base de datos, mediante Hibernate. A su vez, también creamos una clase 'HibernateUtil.java' (aunque puede tener otro nombre) a modo de controlador, para así realizar el proceso de 'establecimiento de la conexión'.

```
<? hibernate.cfg.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC
3     "-//Hibernate/Hibernate Configuration DTD//EN"
4     "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
5
6 <hibernate-configuration>
7     <session-factory>
8         <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
9         <property name="connection.url">jdbc:mysql://localhost:3306/empresa</property>
10        <property name="connection.username">user</property>
11        <property name="connection.password">user4321</property>
12        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
13        <property name="show_sql">true</property>
14        <property name="format_sql">true</property>
15
16        <mapping class="com.kevinzamora.hibernatemysql.model.Emp"></mapping>
17        <mapping class="com.kevinzamora.hibernatemysql.model.Dept"></mapping>
18
19    </session-factory>
20 </hibernate-configuration>
```

```
© HibernateUtil.java x
1 package com.kevinzamora.hibernatemysql.config;
2
3 import org.hibernate.SessionFactory;
4 import org.hibernate.cfg.Configuration;
5
6 public class HibernateUtil { @ 6 usages @ KevinZamora
7
8     private static SessionFactory sessionFactory; @ 3 usages
9
10    public static SessionFactory getSessionFactory() { @ 6 usages @ KevinZamora
11        if (sessionFactory == null) {
12            try {
13                Configuration configuration = new Configuration();
14                configuration.configure( resource: "hibernate.cfg.xml");
15                sessionFactory = configuration.buildSessionFactory();
16            } catch (Exception e) {
17                System.out.println("Error al crear la sesión de Hibernate: " + e.getMessage());
18                throw new ExceptionInInitializerError(e);
19            }
20        }
21        return sessionFactory;
22    }
23
24    public static void shutdown() { sessionFactory.close(); }
25
26 }
```

### 3. Realiza una inserción y un borrado sobre la tabla EMP.

```
public class EmployeeDAO { 3 usages  ▲ KevinZamora
    public void insertEmp(Emp emp) { 1 usage  ▲ KevinZamora
        Transaction transaction = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();
            try {
                session.save(emp);
            } catch (Exception e) {
                System.out.println("Se ha producido el siguiente error al guardar: " + e);
            }
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }

    public void deleteEmp(int empno) { 1 usage  ▲ KevinZamora
        Transaction transaction = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();
            Emp emp = session.get(Emp.class, empno);
            if (emp != null) {
                session.delete(emp);
                transaction.commit();
            } else {
                transaction.rollback();
            }
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }
    }
}
```

Insertando un empleado:  
(Salida mostrada por terminal)

```
Hibernate:
insert
into
    emp
(comm, deptno, hiredate, job, mgr, ename, sal)
values
    (?, ?, ?, ?, ?, ?, ?)
El nuevo empleado se ha guardado correctamente
```

Borrando un empleado:  
(Salida mostrada por terminal)

```
Hibernate:
select
    e1_0.empno,
    e1_0.comm,
    e1_0.deptno,
    e1_0.hiredate,
    e1_0.job,
    e1_0.mgr,
    e1_0.ename,
    e1_0.sal
from
    emp e1_0
where
    e1_0.empno=?
Hibernate:
delete
from
    emp
where
    empno=?
Se ha borrado el empleado creado durante la prueba de inserción
Process finished with exit code 0
```

**Anotación importante:** Para lograr que nos funcione correctamente la inserción de nuevos empleados, ha sido necesario realizar una pequeña modificación sobre el comportamiento de 'la columna (o parámetro) EMPNO' de nuestra tabla 'EMP', alojada en nuestra base de datos 'empresa', para así convertir el citado campo en una variable auto-incremental. Para ello, hemos usado la siguiente sentencia SQL:

```
ALTER TABLE emp MODIFY COLUMN empno INT AUTO_INCREMENT;
```

#### 4. Obtener un listado sobre las tablas EMP y DEPT que visualice empno, ename, sal, dname y loc.

- Código:

```
// Obtener lista de empleados
List<Emp> employees = employeeDAO.listEmps();
List<Dept> departamentos = employeeDAO.listDept();
System.out.println("Lista de empleados:");
for (Emp e : employees) {
    for (Dept d : departamentos) {
        if (d.getDeptno() == e.getDeptno()) {
            System.out.println(e.toString() + ", " + d.toString());
            System.out.println(e.getEmpno() + " - Nombre: " + e.getName() + ", Salario: " + e.getSalary()
                + "€, Departamento: " + d.getName() + ", Localización: " + d.getLocation());
        }
    }
}
```

- Resultado mostrado por consola:

```
Hibernate:
select
  e1_0.empno,
  e1_0.comm,
  e1_0.deptno,
  e1_0.hiredate,
  e1_0.job,
  e1_0.mgr,
  e1_0.ename,
  e1_0.sal
from
  emp e1_0
Hibernate:
select
  d1_0.deptno,
  d1_0.loc,
  d1_0.dname
from
  dept d1_0
Lista de empleados:
7369 - Nombre:SMITH, Salario: 800.0€, Departamento: RESEARCH, Localización: DALLAS
7369 - Nombre:SMITH, Salario: 800.0€, Departamento: RESEARCH, Localización: DALLAS
7499 - Nombre:ALLEN, Salario: 1600.0€, Departamento: SALES, Localización: CHICAGO
7499 - Nombre:ALLEN, Salario: 1600.0€, Departamento: SALES, Localización: CHICAGO
7521 - Nombre:WARD, Salario: 1250.0€, Departamento: SALES, Localización: CHICAGO
7521 - Nombre:WARD, Salario: 1250.0€, Departamento: SALES, Localización: CHICAGO
7566 - Nombre:JONES, Salario: 2975.0€, Departamento: RESEARCH, Localización: DALLAS
7566 - Nombre:JONES, Salario: 2975.0€, Departamento: RESEARCH, Localización: DALLAS
7654 - Nombre:MARTIN, Salario: 1250.0€, Departamento: SALES, Localización: CHICAGO
7654 - Nombre:MARTIN, Salario: 1250.0€, Departamento: SALES, Localización: CHICAGO
7698 - Nombre:BLAKE, Salario: 2850.0€, Departamento: SALES, Localización: CHICAGO
7782 - Nombre:CLARK, Salario: 2450.0€, Departamento: ACCOUNTING, Localización: NEW YORK
7782 - Nombre:CLARK, Salario: 2450.0€, Departamento: ACCOUNTING, Localización: NEW YORK
7788 - Nombre:SCOTT, Salario: 3000.0€, Departamento: RESEARCH, Localización: DALLAS
7788 - Nombre:SCOTT, Salario: 3000.0€, Departamento: RESEARCH, Localización: DALLAS
7839 - Nombre:KING, Salario: 5000.0€, Departamento: ACCOUNTING, Localización: NEW YORK
7839 - Nombre:KING, Salario: 5000.0€, Departamento: ACCOUNTING, Localización: NEW YORK
7844 - Nombre:TURNER, Salario: 1500.0€, Departamento: SALES, Localización: CHICAGO
7844 - Nombre:TURNER, Salario: 1500.0€, Departamento: SALES, Localización: CHICAGO
7876 - Nombre:ADAMS, Salario: 1100.0€, Departamento: RESEARCH, Localización: DALLAS
7876 - Nombre:ADAMS, Salario: 1100.0€, Departamento: RESEARCH, Localización: DALLAS
7900 - Nombre:JAMES, Salario: 950.0€, Departamento: SALES, Localización: CHICAGO
7900 - Nombre:JAMES, Salario: 950.0€, Departamento: SALES, Localización: CHICAGO
7902 - Nombre:FORD, Salario: 3000.0€, Departamento: RESEARCH, Localización: DALLAS
7902 - Nombre:FORD, Salario: 3000.0€, Departamento: RESEARCH, Localización: DALLAS
7934 - Nombre:MILLER, Salario: 1300.0€, Departamento: ACCOUNTING, Localización: NEW YORK
7934 - Nombre:MILLER, Salario: 1300.0€, Departamento: ACCOUNTING, Localización: NEW YORK
7939 - Nombre:Kevin, Salario: 50000.0€, Departamento: RESEARCH, Localización: DALLAS
7939 - Nombre:Kevin, Salario: 50000.0€, Departamento: RESEARCH, Localización: DALLAS
7941 - Nombre:KEVIN, Salario: 50000.0€, Departamento: RESEARCH, Localización: DALLAS
7941 - Nombre:KEVIN, Salario: 50000.0€, Departamento: RESEARCH, Localización: DALLAS
```

#### 5. Redactar un documento donde se explique el proceso seguido para la realización de la práctica. Esta breve memoria trata de ilustrar y representar el proceso seguido para la realización de la tarea propuesta mediante el enunciado anteriormente expuesto.