Práctica No. 3. Operaciones Básicas con Hibernate – Actualizar y Eliminar

En esta práctica se trabajarán las actividades para actualizar y eliminar con llave primaria.

Preparación del Entorno

- Abrir el entorno de desarrollo *Eclipse*
- Descargar de la página web http://academicos.azc.uam.mx/jfg/pags/tarea_taller_web.html el archivo ***Practica03Base.zip*** que ya contiene las clases básicas, archivos de configuración y mapeo
-

Preparación de la Base de Datos

- Descargar de la página web http://academicos.azc.uam.mx/jfg/pags/tarea_taller_web.html del bloque de **Práctica 3**, el archivo ***script_01_generacion_llenado.sql*** el cuál contiene la base de datos y las tablas a utilizar.
- Descargar de la página web http://academicos.azc.uam.mx/jfg/pags/tarea_taller_web.html el archivo ***script_llenado_practica_03.sql*** el cuál contiene los datos a utilizar.
- Abrir el entorno *MySQL Workbench* y ejecutar los *script* para crear la base de datos, las tablas y los datos a utilizar como prueba.

En esta práctica se actualizarán e insertarán elementos utilizando las llaves primarias, en prácticas posteriores se usarán criterios basados en otros atributos (columnas).

A partir de la creación de la base de datos y sus tablas, se tienen las siguientes restricciones:

- No es posible borrar o actualizar una licenciatura si hay alumnos inscritos
- Al borrar un alumno, se borra la relación alumno – uea
- Al borrar una uea, se borra la relación alumno – uea
- Al actualizar un alumno, se actualiza la relación alumno – uea
- Al actualizar una uea, se actualiza la relación alumno – uea

Actualización

Para realizar un actualización se trabaja directamente con el objeto (Clase) a manejar. Esto se consigue con la siguiente instrucción:

```
sesion.get(Nombre.class,id);
```

En donde:

- *sesion* es un objeto tipo ***Session***
- ***Nombre*** representa el nombre de la **clase** de la cuál se quiere actualizar un elemento en la base de datos (**NO de la tabla**)
- ***id*** representa una variable que contiene la ***PK*** del elemento a actualizar

Se recupera este objeto, se actualizan los campos deseados y se manda a actualizar. Al momento de

actualizar los elementos, se deben considerar las posibles restricciones (***Constraints***) que se tengan en la tabla.

Antes de realizar la operación de actualización, es necesario comenzar una Transacción y posteriormente realizar una confirmación (***Commit***) para asegurar que los cambios se reflejen en la base de datos.

```
sesion.beginTransaction();
...
...
sesion.getTransaction().commit();
```

Se cambiará el nombre a una licenciatura.

Principal.java

```
package hibernate.principal;

import hibernate.beans.Licenciatura;
import hibernate.conexion.CrearConexion;
import hibernate.operaciones.Operaciones;

public class Principal {

	public static void main(String[] args) {

		actualizarLicenciaturaPK();
		CrearConexion.closeSessionFactory();

	}

	public static void actualizarLicenciaturaPK(){

		Licenciatura licenciatura = new Licenciatura();
		licenciatura.setNombre("Nuevo Nombre");

		Operaciones operaciones = new Operaciones();
		operaciones.actualizarLicenciatura("101101", licenciatura);

	}

}
```

Operaciones.java

```
public void actualizarLicenciatura(String claveLicenciatura, Licenciatura
licenciatura){

            PropertyConfigurator.configure("logger.properties");

            Session sesion = null;
            try{
                  sesion = CrearConexion.getSessionFactory().openSession();
            }catch(ExceptionInInitializerError ex){
                  System.err.println("No se pudo crear la sesion ");
                throw new ExceptionInInitializerError(ex);
            }

      log.info("ACTUALIZANDO LICENCIATURA " + claveLicenciatura);
                  sesion.beginTransaction();
      Licenciatura licenciaturaRecuperada = sesion.get(Licenciatura.class,
                              claveLicenciatura);
      licenciaturaRecuperada.setNombre(licenciatura.getNombre());
                  sesion.update(licenciaturaRecuperada);
                sesion.getTransaction().commit();
                log.info("SE ACTUALIZÓ LA LICENCIATURA " +
claveLicenciatura);


                  sesion.close();

      }
```

Considerar que por cuestiones de prueba, se pueden enviar a borrar o actualizar datos con llaves primaria que no existan, en este caso, la aplicación finaliza con errores. Se genera una excepción del tipo ***NullPointer***, se atrapará esta excepción y se registrará en el *log*.

Operaciones.java

```
public void actualizarLicenciatura(String claveLicenciatura, Licenciatura
licenciatura){

            PropertyConfigurator.configure("logger.properties");

            Session sesion = null;
            try{
                  sesion = CrearConexion.getSessionFactory().openSession();
            }catch(ExceptionInInitializerError ex){
                  System.err.println("No se pudo crear la sesion ");
                throw new ExceptionInInitializerError(ex);
            }
```

```
                try{
                        log.info("ACTUALIZANDO LICENCIATURA " +
claveLicenciatura);
                        sesion.beginTransaction();
                        Licenciatura licenciaturaRecuperada =
sesion.get(Licenciatura.class,claveLicenciatura);

     licenciaturaRecuperada.setNombre(licenciatura.getNombre());
                        sesion.update(licenciaturaRecuperada);
                    sesion.getTransaction().commit();
                    log.info("SE ACTUALIZÓ LA LICENCIATURA " +
claveLicenciatura);
                }
                catch(NullPointerException e){
     log.error("ERROR AL ACTUALIZAR LA LICENCIATURA " +
claveLicenciatura);
                }finally{
                        sesion.close();
                }
     }
```

El error también se produce cuando se desea actualizar la llave primaria, recordando que según las restricciones de la base de datos, no es posible actualizar o eliminar una licenciatura si esta tiene alumnos inscritos.

Operación de Eliminación

El procedimiento para eliminar un elemento de la base de datos es similar, en este caso no es necesario crear un nuevo objeto, se recupera el alumno a eliminar y luego se elimina.

Operaciones.java

```
      public void eliminarAlumno(String matricula){

            PropertyConfigurator.configure("logger.properties");

            Session sesion=null;
            try{
                  sesion = CrearConexion.getSessionFactory().openSession();
            }catch (Throwable ex) {
            System.err.println("ERROR: No se pudo crear la sesion " + ex);
                throw new ExceptionInInitializerError(ex);
             }
                  log.info("SE ELIMINARÁ AL ALUMNO " + matricula);
                  sesion.beginTransaction();
               Alumno alumnoBorrar = sesion.get(Alumno.class, matricula);
                  sesion.delete(alumnoBorrar);
               sesion.getTransaction().commit();
                  log.info("SE HA ELIMINADO AL ALUMNO " + matricula);

                  sesion.close();
      }
```

En este caso, al eliminar un alumno, se eliminan las *ueas* que había cursado, esto por las restricciones en la tabla.

De manera similar, es posible que se envíe a eliminar un alumno que no existe, produciéndose un error como en la actualización, su manejo es el mismo.

Operaciones.java

```
      public void eliminarAlumno(String matricula){

            PropertyConfigurator.configure("logger.properties");

            Session sesion=null;
            try{
                  sesion = CrearConexion.getSessionFactory().openSession();
            }catch (Throwable ex) {
                System.err.println("ERROR: No se pudo crear la sesion " +
ex);
                throw new ExceptionInInitializerError(ex);
             }
```

```
        try{
              log.info("SE ELIMINARÁ AL ALUMNO " + matricula);
              sesion.beginTransaction();
           Alumno alumnoBorrar = sesion.get(Alumno.class, matricula);
              sesion.delete(alumnoBorrar);
           sesion.getTransaction().commit();
              log.info("SE HA ELIMINADO AL ALUMNO " + matricula);
        }catch (IllegalArgumentException arg){
           log.error("NO SE PUEDE ELIMINAR EL ALUMNO");
        }finally{
              sesion.close();
        }

    }
```