

# ACCESO A DATOS

## UD2: MANEJO DE CONECTORES

### ENUNCIADO 7: Insertar, Borrar, Modificar Departamentos.

Crea un formulario que permita insertar, borrar y modificar departamentos.



A la hora de insertar podrás hacerlo de 2 maneras:

- 1.- Comprueba previamente que el numero de departamento no existe.
- 2.- Realiza la inserción y si falla recoge el error. (1062)

A la hora de modificar podrás hacerlo de 2 maneras:

- 1.- Comprueba previamente que el numero de departamento existe.
- 2.- Realiza la modificación sin realizar comprobación previa. Aquí no hay fallo ya que si no existiera devolvería 0 elementos modificados.

A la hora de borrar (en este caso borrado restringido) podrás hacerlo de 2 maneras:

- 1.- Comprueba previamente que el numero de departamento existe y que no tiene empleados.
- 2.- Realiza el borrado sin realizar comprobación previa y si falla recoge el error (1451).

(Opcionalmente si quieres puedes probar a hacer un borrado en cascada del departamento con empleados). Para realizar esta operativa crea el departamento 100 y 3 empleados en este departamento en la BD. Esto lo realizarás directamente en el SGBD.

# ACCESO A DATOS

## UD2: MANEJO DE CONECTORES

### ENUNCIADO 8: Insertar, Borrar, Modificar Departamentos. Utilizando `prepareStatement()`.

Realiza como mínimo uno de los procesos de insertar, borrar o modificar utilizando consultas por parámetros. (PreparedStatement)

```
String consulta = "insert into departamentos values(?,?,?)";
try ( PreparedStatement sentencia = Pool.getCurrentConexion().prepareStatement(consulta)) {
    //Ahora indicamos lo que es cada parámetro
    sentencia.setInt(1, numdep);
    sentencia.setString(2, nombre);
    sentencia.setString(3, localidad);
    return sentencia.executeUpdate();
} //Esta consulta devuelve el numero de registros afectados si queremos recogerlo. En este caso 1
```

Se aconseja practicar el PreparedStatement cambiando todas las consultas de GestionDepartamento aportadas en la solución del Enunciado 7.

### ENUNCIADO 9: Insertar Empleados

Realiza ahora un proceso que permita insertar empleados. Dado que empleados tiene muchos campos puedes simplificarlo obligando a que introduzca unos pocos campos y el resto lo dejas a null.

Introducirás como mínimo el número de empleado (clave), apellido, y departamento (foránea).

A la hora de introducir la clave foránea podrás elegir 3 opciones (recomendada 2 y 3):

Opción 1: Indicar número departamento.

Opción 2: Indicar nombre departamento.

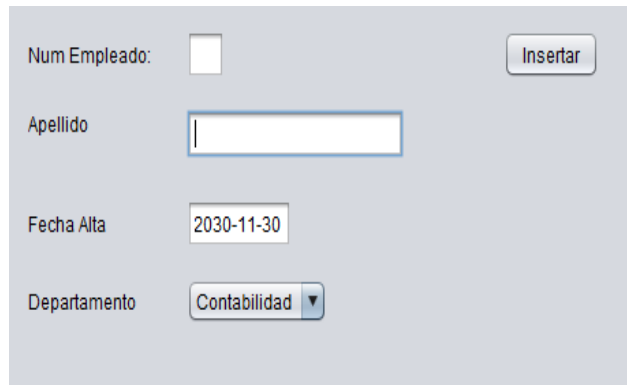
Opción 3: Cargar departamentos en un combo y elegir.

## ACCESO A DATOS

### UD2: MANEJO DE CONECTORES

#### ENUNCIADO 10: Insertar Empleados con fecha de alta

Añade al ejercicio 9 la posibilidad de insertar la fecha de alta del empleado en la empresa. El formato será el de día,mes, año.



(Nota: Observa que lo que vemos en el combo siempre depende del método toString del modelo)

```
@Override
public String toString() {
    // return this.getNombredep()+"--"+this.getLoc(); //Enunciado 4
    //return this.getNum()+" "; //Le añadimos " " para que devuelva el string Enunciado 5
    return this.getNombredep(); // Enunciado 10
}
```

#### ENUNCIADO 11: Realiza un borrado en cascada de un departamento.

Introduce para realizar las pruebas un departamento nuevo (por ejemplo el 100) y crea al menos 2 empleados en este departamento. El introducir datos puedes hacerlo directamente sobre la BD.

Realiza un borrado en cascada del departamento 100, de tal manera que al borrarlo también elimine los empleados que tiene.

*\*Para realizar las pruebas puedes lanzar estos comandos que crear en departamento 100 de nombre borrame y 2 empleados denominados santo y seña \**

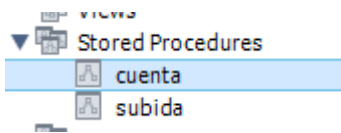
```
INSERT INTO `ejemplo`.`departamentos` (`dept_no`, `dnombre`, `loc`) VALUES ('100', 'Borrame', 'Wirtz');
INSERT INTO `ejemplo`.`empleados` (`emp_no`, `apellido`, `dept_no`) VALUES ('100', 'Santo', '100');
INSERT INTO `ejemplo`.`empleados` (`emp_no`, `apellido`, `dept_no`) VALUES ('200', 'Seña', '100');
```

# ACCESO A DATOS

## UD2: MANEJO DE CONECTORES

### ENUNCIADO 12: Ejercicios con procedimientos almacenados.

Ejecuta el procedimiento almacenado cuenta proporcionado en la BD. Despliega Stored Procedures en el workbench



```
-- -----  
-- Routine DDL  
-- -----  
DELIMITER $$  
  
CREATE PROCEDURE `ejemplo`.`cuenta` (OUT salida int,IN valor int)  
BEGIN  
  select count(*) into salida from empleados where dept_no=valor;  
END$$
```

Observa que tiene el primer parámetro de salida y el segundo de entrada.

El procedimiento lo que hace es recoger en la variable salida todos los empleados de un departamento dado.

Para ejecutar un procedimiento almacenado necesitamos ejecutar un CallableStatement

```
public static void ejecutaproc(int numdep, JLabel lblsalida) throws SQLException {  
    Connection conexion=Pool.getCurrentConexion();  
  
    //independientemente de si el parámetro es de entrada o de salida se pone ?  
    //Cuidado que se abren con llaves.  
    String consulta="{call ejemplo.cuenta(?,?)}";  
  
    try(CallableStatement proc=conexion.prepareCall(consulta)){  
        proc.registerOutParameter(1, Types.INTEGER); //Primer parámetro de salida  
        proc.setInt(2, numdep); //Segundo parámetro de entrada  
        proc.executeQuery(); //Lo ejecutamos  
  
        lblsalida.setText(proc.getInt(1)+""); //Como lo que se recoge es un entero en el parámetro 1  
        //le concatenamos un "" para pasarlo a string  
    }  
}
```

Observa que la ejecución del proc depende del orden y tipo de los parámetros. Puedes probar a cambiar el out salida que sea el parámetro 2 y el valor de entrada el parámetro 1. La ejecución cambiaría.