ACCESO A DATOS
UD2: MANEJO DE CONECTORES

EXPLICACION ENUNCIADO 7: Insertar. No busca previamente su existencia. Directamente va a insertarlo.

```
private void btnInsertarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    //SEGUNDA OPCION: No hacemos la comprobación previa de si existe o no, y
    //recogemos el error que devuelve la BD. En este caso el 1062 es el error de primary key
    if (txtNumero.getText().isEmpty() || txtNombre.getText().isEmpty() || txtLoc.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Faltan Datos");
        return;
    }
    //Desde este try se abre la conexión, por lo que en cualquier via de salida del programa se debe de cerrar
    //Obligamos a que inserte todos los datos
    try {
        JOptionPane.showMessageDialog(null, GestionDepartamento.insertarDepartamento(Integer.valueOf(txtNumero.getText().trim()),
                txtNombre.getText().trim(), txtLoc.getText().trim())
                + " registro insertado");
        Pool.getCurrentConexion().commit();//Ha ido todo bien commit
    } catch (SQLException ex) {
        try {
            switch (ex.getErrorCode()) {
                case 1062 ->
                    JOptionPane.showMessageDialog(null, "Ya existe el departamento");
                default ->
                    Logger.getLogger(Enunciado7_recogiendoerror.class.getName()).log(Level.SEVERE, null, ex);
            }
            Pool.getCurrentConexion().rollback(); //Se ha producido un error rollback
        } catch (SQLException ex1) {
            Logger.getLogger(Enunciado7_recogiendoerror.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Datos incorrectos");
        ponerblancos();
    } finally {
        Pool.Cerrar();
    }
}
```

Pasos:

1.- Inicialmente asegurarnos que se introducen todos los datos. (1)

- Faltan datos: 1 y sale. No hay conexión

2.- Si se introducen todos los datos, pero se introducen mal (ejemplo un numero de departamento abc y no numérico) va al punto (6)

- 1-2-6-7 (En el método pool.cerrar solo cerramos la conexión si se ha abierto). No hay conexión. Parece que abre la conexión en el punto 2 y la cierra en el 8 pero realmente no llega a hacerlo porque salta la excepción al punto 6.

3.- Si se introducen todos los datos va directamente a insertar. Si todo va bien IMPORTANTE COMMIT (3). En el caso de que diera un error la consulta lanzaría la excepción de SQLException y haría un ROLLBACK (5). Tanto haga el commit (todo bien) como el rollback (algo mal) siempre cerramos la conexión (7).

- Proceso correcto.Se introducen datos bien y se ejecuta bien la consulta: 1-2-3-7. Abre la conexión en el punto 2 y lo cierra en el 7. Commit en el 3.
- Proceso incorrecto falla la inserccion.Se introducen datos bien y falla la consulta (este fallo puede ser por ejemplo porque se perdió la conexión a la BD) : 1-2-4(*)-5-7. Abre la conexión en el punto 2 y lo cierra en el 7. Rollback en el 5.

4.- Si se introducen todos los datos y directamente va a insertarlo , pero resulta que el registro ya EXISTE PREVIAMENTE, entonces intentará la inserción y la BD devolverá el error 1062 de que ya existe el departamento. Por último cerramos la conexión.

- 1-2-4(*)-5-8

Gestion de ERRORES (Los errores los devuelve la BD):

IMPORTANTE: Los errores los vamos a calificar de 2 tipos: Error de primary key, es decir, ya existe el registro (Es el error 1062) o cualquier otro error.

```
} catch (SQLException ex) {
    try {
        switch (ex.getErrorCode()) {
            case 1062 ->
                JOptionPane.showMessageDialog(null, "Ya existe el departamento");
            default ->
                Logger.getLogger(Enunciado7_recogiendoerror.class.getName()).log(Level.SEVERE, null, ex);
        }
        Pool.getCurrentConexion().rollback(); //Se ha producido un error rollback
    } catch (SQLException ex1) {
        Logger.getLogger(Enunciado7_recogiendoerror.class.getName()).log(Level.SEVERE, null, ex1);
    }
```

EXPLICACION ENUNCIADO 7: Modificar. No busca previamente su existencia. Directamente va a Modificarlo.

```
private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    /**
     * SEGUNDA OPCION: NO COMPROBAMOS SI EXISTE. El metodo ya nos dirá cero
     * registros afectados en el caso de que modifique un registro que no
     * existe.
     */
    if (txtNumero.getText().isEmpty() || txtNombre.getText().isEmpty() || txtLoc.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Faltan Datos");
        return;
    }
    try {
/           Desde este try se abre la conexión, por lo que en cualquier via de salida del programa se debe de cerrar
/           Obligamos a que inserte todos los datos
        JOptionPane.showMessageDialog(null, GestionDepartamento.ModificarDepartamento(Integer.valueOf(txtNumero.getText().trim()),
                txtNombre.getText().trim(), txtLoc.getText().trim())
                + " registro modificado");
        Pool.getCurrentConexion().commit();//Ha ido todo bien commit
    } catch (SQLException ex) {
        try {
            Pool.getCurrentConexion().rollback(); //Se ha producido un error rollback
        } catch (SQLException ex1) {
            Logger.getLogger(Enunciado7_recogiendoerror.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Datos incorrectos");
        ponerblancos();
    } finally {
        Pool.Cerrar();
    }

}
```

En el caso de mofidicar no recogemos el error porque si existe el registro lo actualiza, y si no existe el registro sería hacer una actualización a un registro no existente, que sería lo mismo que no hacer nada.

Pasos:

1.- Inicialmente asegurarnos que se introducen todos los datos. (1)

- Faltan datos: 1 y sale. No hay conexión

2.- Si se introducen todos los datos, pero se introducen mal (ejemplo un numero de departamento abc y no numérico) va al punto (5)

- 1-2-5-6 (En el método pool.cerrar solo cerramos la conexión si se ha abierto). No hay conexión. Parece que abre la conexión en el punto 2 y la cierra en el 6 pero realmente no llega a hacerlo porque salta la excepción al punto 5.

3.- Si se introducen todos los datos pasa directamente a modificar. En la primera consulta (abre la conexión). Si todo va bien IMPORTANTE COMMIT (2).Aqui no importa si actualiza el registro o no. Hace el commit si se ha ejecutado correctamente la consulta. En el caso de que diera un error la consulta lanzaría la excepción de SQLException y haría un ROLLBACK (4). Tanto haga el commit (todo bien) como el rollback (algo mal) siempre cerramos la conexión (6).

- Proceso correcto.Se introducen datos bien : 1-2-3-6. Abre la conexión en el punto 2 y lo cierra en el 6. Commit en el 3.
- Proceso incorrecto falla la modificación.Se introducen datos bien y existe el departamento: 1-2-4-6. Abre la conexión en el punto 2 y lo cierra en el 6. Rollback en el 4.

EXPLICACION ENUNCIADO 7: Borrar restringido. No busca previamente si tiene empleados. Directamente va a borrarlo. Si tiene empleados da error de foreing key en la tabla empleados (1467).

```java
private void btnBorrarActionPerformed(java.awt.event.ActionEvent evt) {

    //En este caso vamos a hacer un borrado restringido. De tal manera que si existen empleados
    //No podemos realizar el borrado.

    //SEGUNDA OPCION: No hacemos la comprobación previa de si existe o no, y
    //recogemos el error que devuelve la BD. En este caso el 1451 de foreign key
    if (txtNumero.getText().isEmpty() || txtNombre.getText().isEmpty() || txtLoc.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Faltan Datos");
        return;
    }
    try {
        //Desde este try se abre la conexión, por lo que en cualquier via de salida del programa se debe de cerrar
        //Obligamos a que inserte todos los datos
        JOptionPane.showMessageDialog(null, GestionDepartamento.borrarDepartamento(Integer.valueOf(txtNumero.getText().trim())))
                + " registro borrado");
        Pool.getCurrentConexion().commit();//Ha ido todo bien commit
        ponerblancos();
    } catch (SQLException ex) {
        try {
            //System.out.println(ex.getErrorCode()); Descoméntalo si quieres verificar el número de error
            switch (ex.getErrorCode()) {
                case 1451 ->
                    JOptionPane.showMessageDialog(null, "El departamento tiene empleados");
                default ->
                    Logger.getLogger(Enunciado7_recogiendoerror.class.getName()).log(Level.SEVERE, null, ex);
            }
            Pool.getCurrentConexion().rollback(); //Se ha producido un error rollback
        } catch (SQLException ex1) {
            Logger.getLogger(Enunciado7_recogiendoerror.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Datos incorrectos");
        ponerblancos();
    }finally{
        Pool.Cerrar();
    }
}
```

Pasos:

1.- Inicialmente asegurarnos que se introducen todos los datos. (1)

- Faltan datos: 1 y sale. No hay conexión

2.- Si se introducen todos los datos, pero se introducen mal (ejemplo un numero de departamento abc y no numérico) va al punto (6)

- 1-2-6-7 (En el método pool.cerrar solo cerramos la conexión si se ha abierto). No hay conexión. Parece que abre la conexión en el punto 2 y la cierra en el 7 pero realmente no llega a hacerlo porque salta la excepción al punto 6.

3.- Si se introducen todos los datos y bien, va directamente a borrarlo. Si todo va bien IMPORTANTE COMMIT (6). En el caso de que diera un error la consulta lanzaría la excepción de SQLException y haría un ROLLBACK (7). El error será de 2 tipos bien la consulta incorrecta, bien que no puede borrarlo porque no pueden quedar empleados sin departamento (error 1451). Tanto haga el commit (todo bien) como el rollback (algo mal) siempre cerramos la conexión (7).

- Proceso correcto.Se introducen datos bien y borramos de manera correcta porque el departamento no tiene empleados. 1-2-3-7 Abre la conexión en el punto 2 y lo cierra en el 7. Commit en el 3.

- Se introducen bien los datos , pero el departamento tiene empleados, o se produce cualquier error en la consulta. 1-2-4-5-7. Abre la conexión en el punto 2 y lo cierra en el 7.
- Ten en cuenta que borrar un departamento no existente no produce ningún error, sencillamente no borra nada.