Diseño y Desarrollo de Sistemas de Información

Llamadas a procedimientos y funciones almacenadas mediante JDBC

Procedimientos y funciones almacenadas

- En el desarrollo de aplicaciones es conveniente, siempre que se pueda, que la lógica de negocio se implemente en el propio SGBD. Una de las formas habituales de hacerlo es mediante procedimientos, funciones y disparadores
- Estos procedimientos se pueden utilizar, entre otras cosas, para realizar cálculos y operaciones complejas
- Una buena práctica consiste en programar y lanzar excepciones desde los procedimientos almacenados cuando se detecta un error de lógica de negocio
- La excepción debe enviar el mensaje de error para que sea capturado en las capas superiores

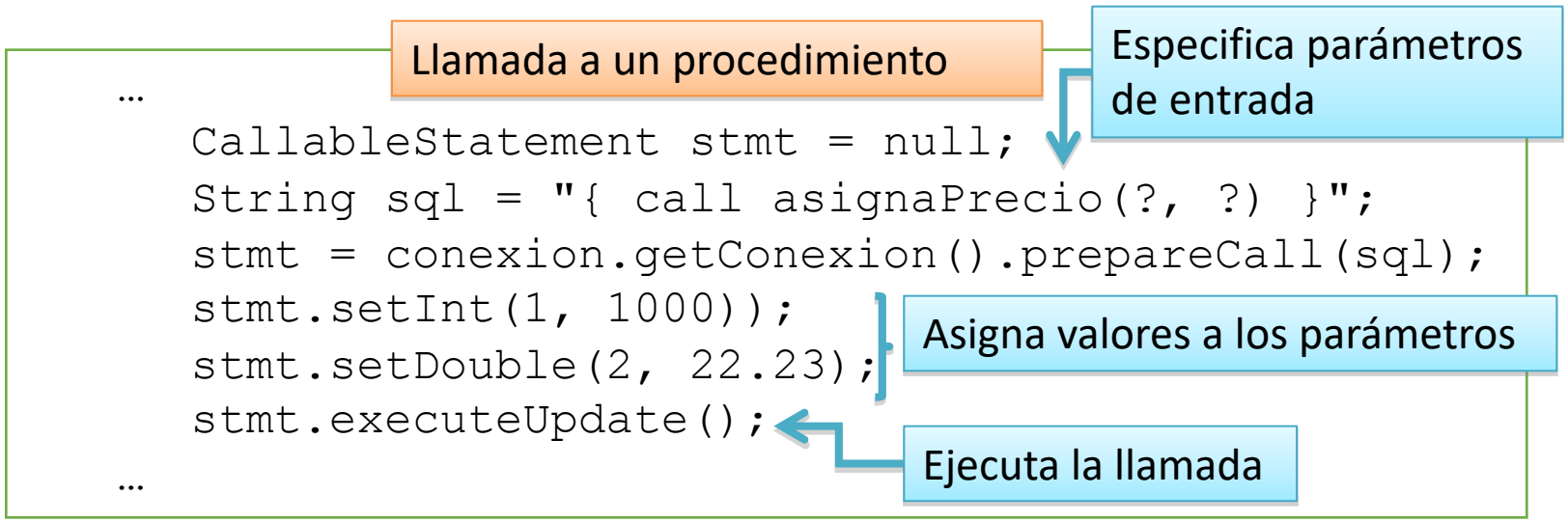
```
BEGIN
    ...
   IF vNumeroOrdenadores > vMaximoOrdenadores THEN
      RAISE_APPLICATION_ERROR (-20001, 'El aula ' || pAula || 'tiene ' ||
                                  vNumeroOrdenadores || ' ordenadores');
   END IF;
   ...
END;
```

getErrorCode()

getMessage()

- La llamada a un procedimiento almacenado se realiza con un objeto de la clase CallableStatement, que recibe el resultado del método prepareCall() de la clase Connection
- La llamada al procedimiento o función se prepara previamente utilizando el símbolo ? para indicar la posición de los parámetros

- Para asignar valores a los parámetros de entrada se utiliza el método setXXX(), donde XXX indica el tipo de datos del parámetro de entrada
- Finalmente se llama al método executeUpdate() para ejecutar la llamada al procedimiento o función



- Los procedimientos almacenados pueden tener parámetros de entrada y parámetros de salida, que se utilizan para devolver información al programa. Específicamente, las funciones siempre tienen un parámetro de salida, que es el valor que devuelve dicha función
- Para que un procedimiento o una función devuelva información mediante los parámetros de salida es necesario indicar, previamente, de qué tipo es cada parámetro mediante el método registerOutParameter()

- Para recuperar los valores de los parámetros de salida se utiliza el método getXXX(), donde XXX indica el tipo de datos del parámetro de salida

Llamada a una función

```
…
 CallableStatement stmt = null;
 String sql = "{ ? = call asignaPrecio(?) }";
 stmt = conexion.getConexion().prepareCall(sql);
 stmt.setInt(2, 30));
 stmt.registerOutParameter(1, Types.DOUBLE);
 stmt.executeUpdate();
 double resultado = stmt.getDouble(1);
…
```

Especifica el parámetro de salida

Uso de cursores como parámetro de salida

Fragmento de código Java para realizar una llamada al procedimiento almacenado

```
private static void callOracleStoredProcCURSORParameter()
                                        throws SQLException {

String getUserCursorSql = " { call getUserCursor (?, ?) } ";
ResultSet rs = null;
try {
    CallableStatement call =
        conexion.getConexion().prepareCall (getUserCursorSql);
    call.setString(1, "Juan");
    call.registerOutParameter(2, OracleTypes.CURSOR);
    call.executeUpdate();

    rs = (ResultSet) call.getObject(2);

    while (rs.next()) {
      System.out.println("ID: " + rs.getString(1));
      System.out.println("Nombre: " + rs.getString(2));
      System.out.println("Fecha de nacimiento: " + rs.getString(3));
      }
    } catch (SQLException e) {
            System.out.println(e.getMessage());
    } finally {
            if (rs != null) rs.close();
            if (call != null) call.close();
    }
}
```

Método *get* para capturar un cursor

Código PL/SQL del procedimiento almacenado

```
CREATE OR REPLACE PROCEDURE getUserCursor (
        pUsername IN DBUSER.USERNAME%TYPE,
        cUser OUT SYS_REFCURSOR)
IS
BEGIN

 OPEN cUser FOR
 SELECT * FROM USERS
      WHERE USERNAME LIKE pUsername || '%';

END;
```