2.10 EJECUCIÓN DE PROCEDIMIENTOS.

Los procedimientos almacenados en la base de datos consisten en un conjunto de sentencias SQL y del lenguaje procedural utilizado por el sistema gestor de base de datos que se pueden llamar por su nombre para llevar a cabo alguna tarea en la base de datos.

Pueden definirse con parámetros (o también argumentos) de entrada (IN),

de salida (OUT),

de entrada/salida (INOUT)

o sin ningún parámetro.

También pueden devolver un valor, en este caso se trataría de una función.

Las técnicas para desarrollar procedimientos y funciones almacenadas dependen del sistema gestor de base de datos, **en MySQL por ejemplo las funciones no admiten parámetros OUT e INOUT, sólo admiten parámetros IN.**

EJEMPLOS:

Procedimiento que sube el salario de los empleados de un departamento. Se reciben dos parámetros del departamento (d) y la subida (subida). Se llama *subida_sal:*

Procedimiento en ORACLE:

```
CREATE OR REPLACE PROCEDURE subida_sal(d NUMBER, subida NUMBER) AS
BEGIN
   UPDATE empleados SET salario = salario + subida WHERE dept_no = d;
   COMMIT;
END;
/
```

Procedimiento en MySQL:

```
delimiter //
CREATE PROCEDURE subida_sal(d INT, subida INT)
BEGIN
   UPDATE empleados SET salario = salario + subida WHERE dept_no = d;
   COMMIT;
END;
//
```

```
delimiter **
CREATE PROCEDURE subida_sal(d INT, subida INT)
BEGIN
   UPDATE empleados SET salario = salario + subida WHERE dept_no = d;
   COMMIT;
END;
**
```

Función (en ORACLE) de nombre *nombre_dep* con dos parámetros, el primero es de entrada y recibe un número de departamento, el segundo es de salida, se utilizará para guardar la localidad del departamento; la función devuelve el nombre del departamento.

Si el departamento no existe devuelve como nombre *"INEXISTENTE"*:

Función en ORACLE:

```
CREATE OR REPLACE FUNCTION nombre_dep
                (d NUMBER, locali OUT VARCHAR2)  RETURN VARCHAR2 AS
     nom VARCHAR2(15);
 BEGIN
     SELECT dnombre, loc INTO nom, locali FROM departamentos
     WHERE dept_no = d;
     RETURN nom;
 EXCEPTION
     WHEN NO_DATA_FOUND THEN
        nom := 'INEXISTENTE';
        RETURN nom;
 END;
/
```

El siguiente ejemplo crea una función (en MySQL) de nombre *nombre_dep*, recibe un número de departamento (parámetro de entrada) y devuelve el nombre si existe; si no existe devuelve como nombre *"INEXISTENTE"*:

```
DELIMITER //
 CREATE FUNCTION nombre_dep(d int) RETURNS VARCHAR(15)
 BEGIN
   DECLARE nom VARCHAR(15);
   SET nom = 'INEXISTENTE';
   SELECT dnombre INTO nom FROM departamentos
   WHERE dept_no=d;
   RETURN nom;
 END;
//
```

Para ejecutarlo desde MySQL escribimos: `SELECT nombre_dep(10);`

A continuación, se muestra un procedimiento (en MySQL) que recibe un número de departamento y devuelve en forma de parámetros de salida el nombre y la localidad (las **funciones** no pueden usar parámetros OUT pero las **procedures** si), se asigna un valor inicial a los parámetros de salida por si el departamento no existe:

```
DELIMITER //
 CREATE PROCEDURE datos_dep
           (d int, OUT nom VARCHAR(15), OUT locali VARCHAR(15))
 BEGIN
   SET locali = 'INEXISTENTE';
   SET nom = 'INEXISTENTE';
   SELECT dnombre, loc INTO nom, locali FROM departamentos
   WHERE dept_no=d;
 END;
//
```

Para ejecutarlo desde MySQL escribimos las siguientes sentencias:

```
CALL datos_dep(10, @nom, @locali);
SELECT @nom, @locali;
```

La interfaz **CallableStatement** permite que se pueda llamar desde Java a los procedimientos almacenados.

Para crear un objeto se llama al método ***prepareCall(String)*** del objeto **Connection**.

En el *String* se declara la llamada al procedimiento o función, tiene dos formatos, uno incluye el parámetro de resultado (usado para las funciones) y el otro no:

Función

```
{? = call <nombre_procedure>[(<arg1>,<arg2>, ...)]}
```

Procedimiento o procedure

```
{call <nombre_procedure>[(<arg1>,<arg2>, ...)]}
```

Si los procedimientos y funciones incluyen parámetros de entrada o de salida es necesario indicarlos en forma de marcadores de posición.

La referencia a los parámetros es secuencial, por número, el primer parámetro es el 1, el siguiente el 2, etc.

El parámetro de resultado y los parámetros de salida deben ser registrados antes de realizar la llamada.

El siguiente ejemplo declara la llamada al procedimiento ***subida_sal*** que tiene dos parámetros de entrada, se usan los marcadores de posición (?) para indicarlo:

```
String sql= "{ call subida_sal (?, ?) } ";
CallableStatement llamada = conexion.prepareCall(sql);
```

Hay 4 formas de declarar las llamadas a los procedimientos y funciones que dependen del uso u omisión de parámetros, y de la devolución de valores. Son las siguientes:

- **{ call nombre_procedimiento}**: para un procedimiento almacenado sin parámetros.
- **{ ? = call nombre_función }**: para una función almacenada que devuelve un valor y no recibe parámetros, el valor se recibe a la izquierda del igual y es el primer parámetro llamado parámetro de resultado.
- **{ call nombre_procedimiento(?, ?, ...) }**: para un procedimiento almacenado que recibe parámetros.
- **{ ? = call nombre_función(?, ?, ...) }**: para una función almacenada que devuelve un valor (primer parámetro) y recibe varios parámetros.

En el siguiente ejemplo se realiza una llamada al procedimiento *subida_sal* (de MySQL); los valores de los parámetros se asignan a partir de los argumentos de *main()*:

```
import java.sql.*;
public class ProcSubida {
   public static void main(String[] args) {
     try {
          Class.forName("com.mysql.jdbc.Driver");
          Connection conexion = DriverManager.getConnection
           ("jdbc:mysql://localhost/ejemplo", "ejemplo", "ejemplo");

          //recuperar parámetros de main
          String dep = args[0];    //departamento
          String subida = args[1]; //subida

          //construir orden de llamada
          String sql= "{ call subida_sal (?, ?) } ";

          //Preparar la llamada
          CallableStatement llamada = conexion.prepareCall(sql);

          //Dar valor a los argumentos
```

```
            llamada.setInt(1,Integer.parseInt(dep));       //primero
            llamada.setFloat(2,Float.parseFloat(subida)); // segundo

            //Ejecutar el procedimiento
            llamada.executeUpdate();
            System.out.println ("Subida realizada....");

            llamada.close();
            conexion.close();
        }
        catch (ClassNotFoundException cn) { cn.printStackTrace(); }
        catch (SQLException e)            { e.printStackTrace(); }

    }//fin de main
}//fin de la clase
```

La ejecución desde la línea de comandos y suponiendo que el conector MySQL está en el CLASSPATH visualiza la siguiente información:

```
java ProcSubida 30 200
Subida realizada....
```

En MySQL al ejecutarlo puede que se muestre el siguiente error: ***java.sql.SQLException: User does not have access to metadata required to determine stored procedure parameter types*** ... si el usuario no tiene permisos para ejecutar procedimientos.

En este caso debemos darle el privilegio SELECT sobre la tabla de sistema **mysql.proc** que contiene la información sobre todos los procedimientos almacenados en la base de datos;

se ejecutaría la siguiente orden desde la línea de comandos de MySQL o desde el entorno gráfico que usemos:

GRANT SELECT ON mysql.proc TO 'ejemplo2020'@'localhost';

Los parámetros de salida (**OUT**) **deben ser registrados antes de que la llamada tenga lugar**.

El método que se utilizará es: ***registerOutParameter(int índice, int tipoSQL)***, el primer parámetro es la posición y el siguiente es una constante definida en la clase **java.sql.Types**.

ESTOS SON LOS TIPOS: *ARRAY, BIGINT, BINARY, BIT, BLOB, BOOLEAN, CHAR, CLOB, DATALINK, DATE, DECIMAL, DISTINCT,DOUBLE, FLOAT, INTEGER, JAVA_OBJECT, LONGNVARCHAR, LONGVARBINARY, LONGVARCHAR, NCHAR, NCLOB, NULL, NUMERIC, NVARCHAR, OTHER, REAL, REF, REF_CURSOR, ROWID, SMALLINT, SQLXML, STRUCT, TIME, TIME_WITH_TIMEZONE, TIMESTAMP, TIMESTAMP_WITH_TIMEZONE, TINYINT, VARBINARY, VARCHAR*

En el ejemplo es de tipo VARCHAR, en la llamada al método escribimos lo siguiente:

```
llamada.registerOutParameter(2, java.sql.Types.VARCHAR);
```

Una vez ejecutada la llamada al procedimiento, los valores de los parámetros OUT e INOUT se obtienen con los métodos ***getXXX()*** similares a los utilizados en un **ResultSet**.

El siguiente ejemplo ejecuta el procedimiento *nombre_dep* (de Oracle); desde los argumentos de *main()* se recibe el número de departamento cuyos datos se visualizarán:

```
import java.sql.*;
public class FuncNombre {
```

```
   public static void main(String[] args) {
     try {
       Class.forName("oracle.jdbc.driver.OracleDriver");
       Connection conexion = DriverManager.getConnection
        ("jdbc:oracle:thin:@localhost:1521:XE", "ejemplo", "ejemplo");

       //recuperar parametro de main
       String dep = args[0]; //departamento

       //Construir orden de llamada
       String sql = "{ ? = call nombre_dep (?, ?) } "; // ORACLE

       //Preparar la llamada
       CallableStatement llamada = conexion.prepareCall(sql);

       //registrar parámetro de resultado
       llamada.registerOutParameter(1, Types.VARCHAR);//valor devuelto

       llamada.setInt(2,Integer.parseInt(dep));      //param de entrada

       //Registrar parámetro de salida
       llamada.registerOutParameter(3, Types.VARCHAR);//parámetro OUT

       //Ejecutar el procedimiento
       llamada.executeUpdate();
       System.out.printf("Nombre Dep: %s, Localidad: %s %n",
                         llamada.getString(1), llamada.getString(3));
       llamada.close();
       conexion.close();
     }
     catch (ClassNotFoundException cn) { cn.printStackTrace(); }
     catch (SQLException e)            { e.printStackTrace(); }
   }// fin de main
}// fin de la clase
```

Actividad 2.12

Crea una **función en Oracle**, que reciba un número de departamento y devuelva el salario medio de los empleados de ese departamento y como parámetro de salida el número de empleados.

Si el departamento no existe debe devolver como salario medio el valor -1 y el número de empleados será 0. Si sí existe y no tiene empleados debe devolver 0.

Realiza después un programa Java que use dicha función. El programa recorrerá la tabla *departamentos* y mostrará los datos del departamento, incluyendo el número de empleados y el salario medio. Para cada departamento se realizará una llamada a la función de Oracle.

Realiza un **procedimiento en MySQL** que funcione de forma similar a la función en Oracle, es decir debe recibir un número de departamento y como parámetros de salida debe devolver el número de empleados y el salario medio. Realiza después un programa Java para usar dicho procedimiento, igual que antes el programa recorrerá la tabla *departamentos* y mostrará los datos del departamento, incluyendo el número de empleados y el salario medio.

La función y el procedimiento se crearán desde un programa Java.

SE PUEDE UTILIZAR StringBuilder para la orden

Por ejemplo para crear una vista:

```
StringBuilder sql = new StringBuilder();
sql.append("CREATE OR REPLACE VIEW totales ");
sql.append("(dep, dnombre, nemp, media) AS ");
sql.append("SELECT d.dept_no, dnombre, COUNT(emp_no), AVG(salario) ");
sql.append("FROM departamentos d LEFT JOIN empleados e " );
sql.append("ON e.dept_no = d.dept_no ");
sql.append("GROUP BY d.dept_no, dnombre ");
System.out.println(sql);

Statement sentencia = conexion.createStatement();
int filas = sentencia.executeUpdate(sql.toString());
System.out.printf("Resultado  de la ejecución: %d %n", filas);
```

```
--EN ORACLE SE CREA LA FUNCIÓN ASÍ-----------------------------------------
CREATE OR REPLACE FUNCTION FACTIVIDAD12 (d NUMBER, num out number)
 RETURN number AS
 media number;
 C NUMBER;
BEGIN
 --existe el dep
 SELECT COUNT(*) INTO C FROM DEPARTAMENTOS WHERE DEPT_NO=d;
 IF C = 0 THEN
   media :=-1;
   num:=0;
 ELSE
     SELECT nvl(AVG(SALARIO),0), count(emp_no)
        INTO media, num
     FROM empleados WHERE dept_no=d;
 END IF;
 RETURN media;
END;
 /
```

Haciendo uso de la función FACTIVIDAD12, obtener el siguiente listado:

```
DEPT-NO      NOMBRE     LOCALIDAD   MEDIASAALRIO   CONTADOREMPLES
-------      ---------  ---------   ------------   ----------------
 Xxx         xxxxxxxx   xxxxxxxx       xxxx           xxxxx
 Xxx         xxxxxxxx   xxxxxxxx       xxxx           xxxxx
-------      ---------  ---------   ------------   ----------------
 TOTALES:                            xxxxxxxxx       xxxxxxxxxx
```

```
--PARA PROBARLA
DECLARE
  D NUMBER;
  NUM NUMBER;
  MEDIA NUMBER;
 BEGIN
   MEDIA:= FACTIVIDAD12(41,NUM);
   DBMS_OUTPUT.PUT_LINE(MEDIA);
   DBMS_OUTPUT.PUT_LINE(NUM);
END;
 /

 --EN MYSQL SE CREA EL PROCEDIMIENTO ASÍ----------------------------------
DELIMITER //
CREATE PROCEDURE FACTIVIDAD12 (d int, OUT MEDIA FLOAT, OUT NUM INT)
```

```
 BEGIN
   DECLARE C INT;

   SET MEDIA=0;
   SET NUM = 0;

   SELECT COUNT(*) INTO C FROM DEPARTAMENTOS WHERE DEPT_NO=d;
   IF C = 0 THEN
     SET media =-1;
   ELSE
     SELECT COALESCE(AVG(SALARIO),0), count(emp_no)
        INTO media, num
     FROM empleados WHERE dept_no = d;
   END IF;
  END;
//
```

***Desde MySQL la pruebo:**

```
CALL FACTIVIDAD12(10, @MEDIA, @NUM);
SELECT @MEDIA;
SELECT @NUM;
```

***Dar privilegios al usuario**

```
*--GRANT SELECT ON mysql.proc TO 'ejemplo'@'localhost';--
```