

# LENGUAJE TRANSACCIONAL EN SQL

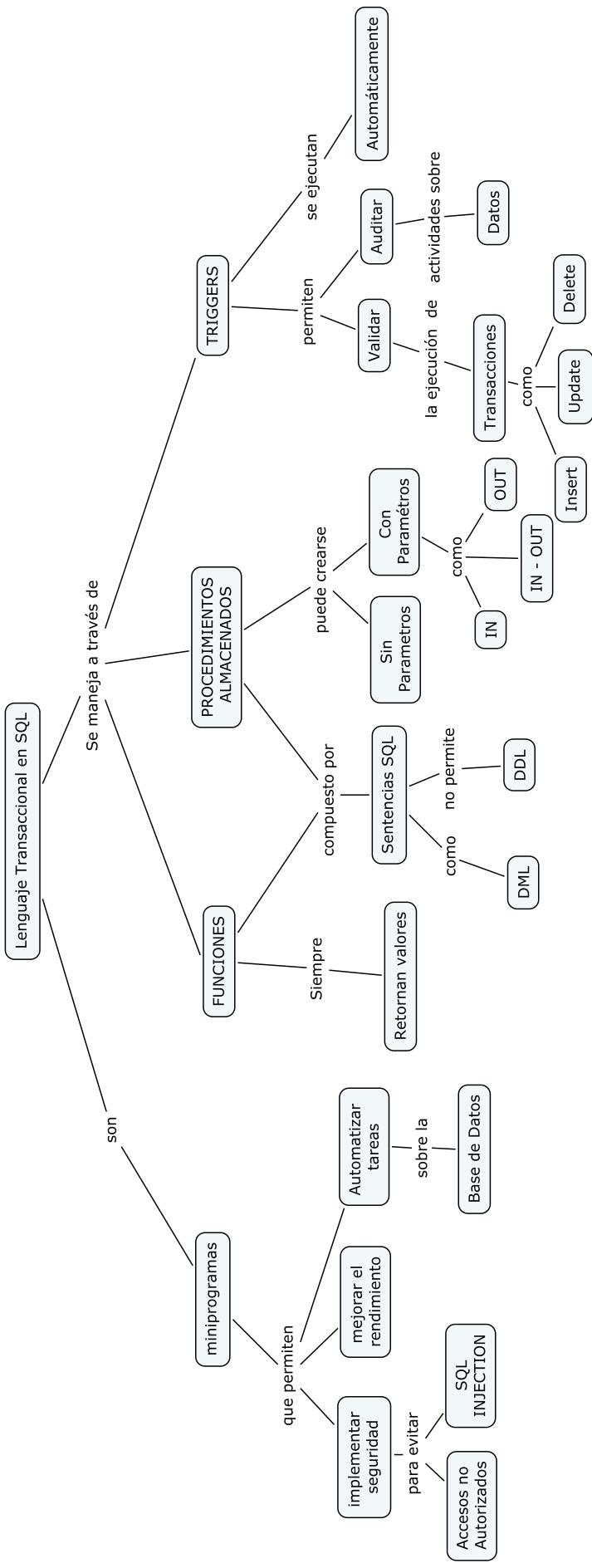
---

INTRODUCCIÓN .....	3
1. PROCEDIMIENTOS ALMACENADOS .....	3
1.1. Creación de procedimientos Almacenados SIN parámetros en MySQL .....	6
1.2. Procedimientos almacenados con parámetros .....	9
2. FUNCIONES .....	19
2.1. FUNCIONES EN MYSQL .....	19
2.2. Funciones en Oracle .....	20
3. TRIGGERS .....	21
3.1. CREACION DE UN TRIGGER EN MySQL.....	21
3.2. BORRADO DE UN TRIGGER EN MySQL .....	26
3.3. CREACION DE UN TRIGGER EN Oracle .....	27
3.4. BORRADO DE UN TRIGGER EN Oracle .....	33



# Mapa conceptual LENGUAJE TRANSACCIONAL EN SQL

2



## INTRODUCCIÓN

Después de haber realizado las respectivas fases de análisis y construcción de bases de datos, llega el momento de sistematizar algunos procesos para el manejo de las diferentes operaciones en una base de datos. En este objeto de aprendizaje se trabajará en los fundamentos para aplicar el lenguaje transaccional en la implementación de funcionalidades en el SGBD.

Se centrará la atención en la construcción de programas usando lenguaje transaccional SQL, con el fin de proporcionar funcionalidades que son implementadas en la base de datos y utilizadas por la capa de datos.

Las implementaciones más comunes son los procedimientos almacenados (stored procedures), las funciones (functions) y los desencadenadores (triggers), en este objeto todos estos serán referenciados para los Sistemas de gestión de Base de datos MySQL y Oracle.

### 1. PROCEDIMIENTOS ALMACENADOS

Un procedimiento almacenado es un conjunto de comandos SQL que pueden almacenarse en el servidor. Una vez que se hace, los clientes no necesitan relanzar los comandos individuales pero pueden en su lugar referirse al procedimiento almacenado.<sup>1</sup>, en este primer momento se va a trabajar desde MySQL.

Son subprogramas que ejecutan una acción específica y no devuelven ningún valor. Tienen un nombre, un conjunto de parámetros (opcional) y un bloque de código.

Una vez se crean los procedimientos almacenados, el sistema no tiene necesidad de verificar la sintaxis de la instrucción SQL, porque con solo utilizar el nombre del procedimiento, se ejecuta la instrucción.

Los procedimientos almacenados permiten implementar elementos de seguridad a las aplicaciones, ya que evitan la manipulación directa de los datos, los usuarios de los procedimientos deben seguir procesos de autenticación y no importaría el lenguaje o plataforma de acceso.

---

<sup>1</sup> <http://dev.mysql.com/doc/refman/5.0/es/stored-procedures.html> (05/08/2013)

Por otro lado mejoran el rendimiento en los clientes, ya que se envía menos información al servidor, aunque aumenta la carga del servidor.

La estructura de un procedimiento en general consta de un nombre, la lista de parámetros en caso de requerirlos y el cuerpo del procedimiento (definición).

CREATE  
PROCEDURE

Nombre  
Procedimiento

(Parámetros)

## Instrucciones SQL

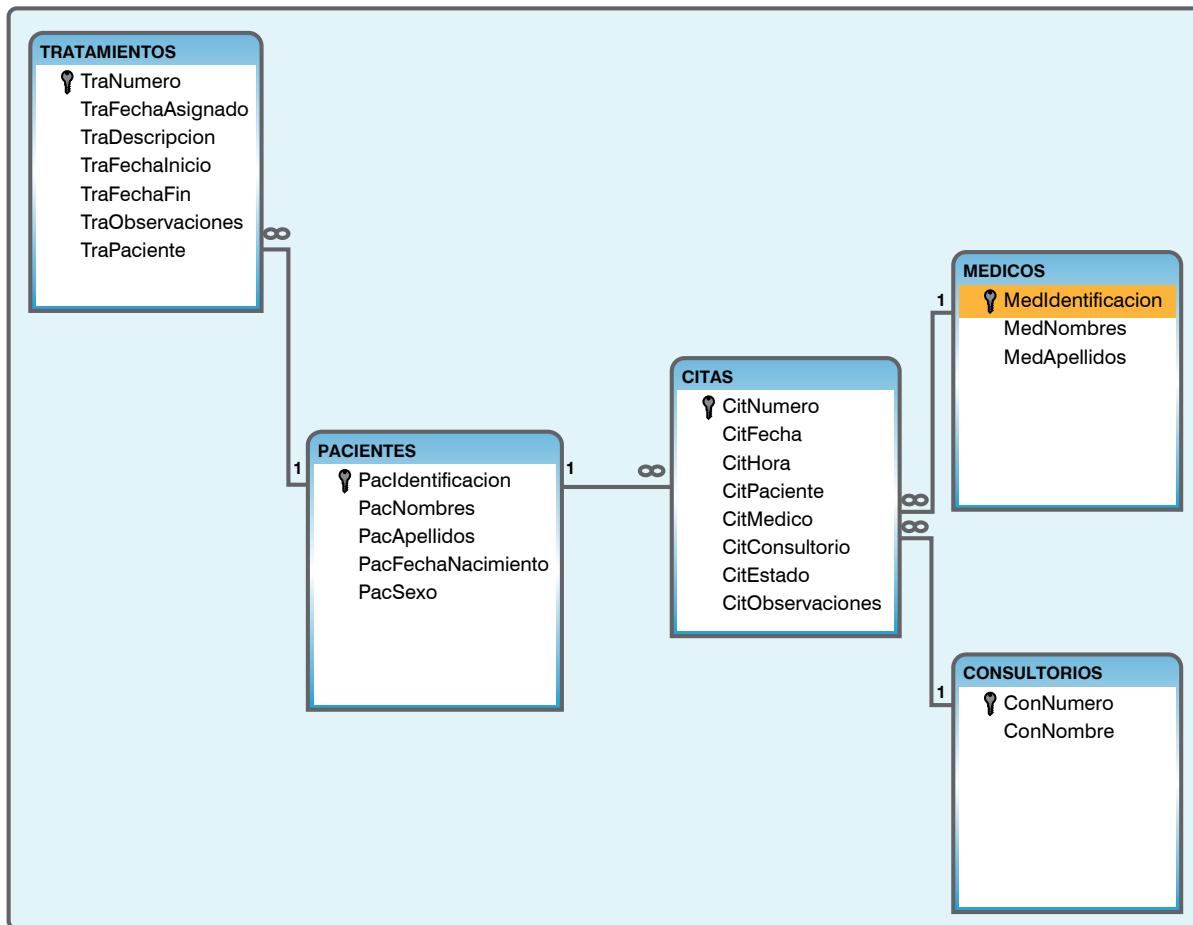
Por cada uno de los parámetros que se requieran se debe hacer la siguiente especificación:

**[ IN | OUT | INOUT ] Nombre\_del\_parametro tipo\_de\_dato**

Algunas de las palabras reservadas a utilizar en la construcción del procedimiento almacenado pueden ser:

- IN : Nos indica que el parámetro será de entrada
- OUT : Nos indica que el parámetro será de salida
- INTOUT : Indica que el parámetro será tanto de Entrada como de salida
- BEGIN: Limitador del procedimiento
- END: Fin de nuestro procedimiento
- DELIMITER: Restablece el punto y coma como delimitador.
- CALL: para llamar al procedimiento una vez creado

La sintaxis y palabras reservadas pueden variar dependiendo del Sistema de Gestión de Base de Datos donde se implemente, en este material se expondrá su construcción para MySQL y Oracle, las instrucciones que van al interior del procedimiento son sentencias SQL acompañadas de algunos elementos del lenguaje de programación tales como variables, inicio y fin de bloques, estructuras de control y repetitivas.



Nota: Para los ejemplos presentados en esta plantilla se utilizará la base de datos del consultorio médico tratada en materiales anteriores. A continuación se presenta el modelo relacional y el diccionario de datos.

Con su respectivo diccionario de Datos:

Tabla	Atributo – Campo	Tipo de Dato	Longitud	Modificador	Tabla y campo foránea
Pacientes	PacIdentificacion	char	10	primary key, not null	
Pacientes	PacNombres	Varchar	50	not null	
Pacientes	PacApellidos	Varchar	50	not null	
Pacientes	PacFechaNacimiento	Date		not null	
Pacientes	PacSexo	Por tener el Modificador enum, no se declara		(ENUM('M', 'F'))	
Medicos	MedIdentificacion	Char	10	primary key, not null	
Medicos	MedNombres	Varchar	50	not null	
Medicos	MedApellidos	Varchar	50	not null	
Consultorios	ConNumero	Int	3	primary key, not null	
Consultorios	ConNombre	Varchar	50	not null	
Tratamientos	TraNumero	Int		primary key, auto_increment	
Tratamientos	TraFechaAsignado	Date		not null	
Tratamientos	TraDescripcion	Text		not null	
Tratamientos	TraFechaInicio	Date		not null	
Tratamientos	TraFechaFin	Date		not null	
Tratamientos	TraObservaciones	Text		not null	
Tratamientos	TraPaciente	Char	10	not null	Pacientes (PacIdentificacion)

### 1.1. Creación de procedimientos Almacenados SIN parámetros en MySQL

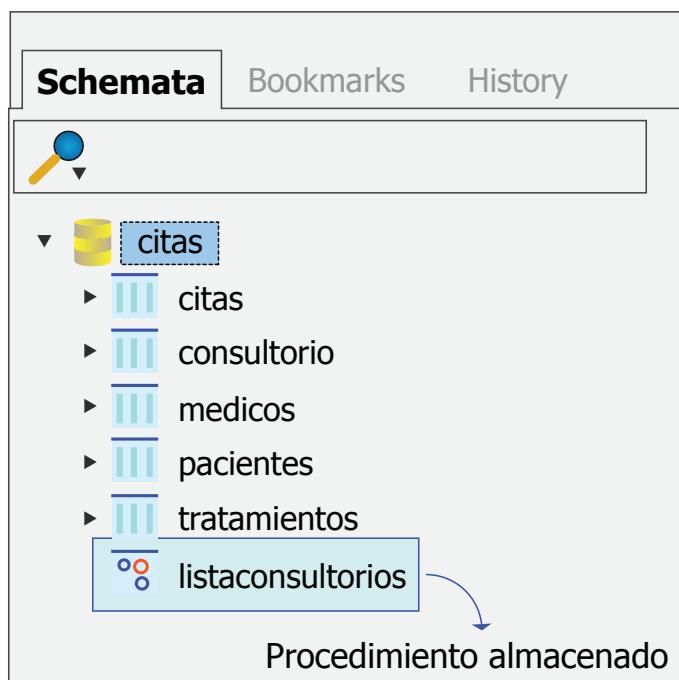
Los procedimientos almacenados y funciones son nuevas funcionalidades de la versión de MySQL 5.0. sigue la sintaxis de SQL:2003 para procedimientos almacenados.

Los procedimientos quedan implementados como objetos de las base de datos, por eso para su creación y ejecución se debe estar dentro del contexto de la base de datos y tener permisos para su creación.

El siguiente ejemplo, ilustra un procedimiento básico, que consta de una sola sentencia SQL. El procedimiento almacenado llamado listarconsultorios cuyo objetivo es listar todos los datos de la tabla consultorios se presenta a continuación:

Use citas Se informa la base de datos en la que se va a trabajar  
create procedure listaconsultorios() Se informa que se va a crear el procedimiento,  
select\*from consultorios; Llamado listaconsultorios  
Instrucción SQL, para la consulta

Al dar enter en el punto y coma de la instrucción, el procedure queda almacenado automáticamente en el esquema de la base de datos.



Una vez creado el procedimiento almacenado listaconsultorios, se puede ejecutar, cada vez que necesitemos visualizar los consultorios, LA VENTAJA de llamar (ejecutar) el procedimiento, es que el sistema omite la revisión de la sintaxis SQL, situación que no sucede cuando se tiene el código SQL sin procedimiento almacenado.

Para llamar el procedimiento almacenado, se utiliza la instrucción:

```
call <nombre_procedimiento>()
```

para el ejemplo presentado: **call listaconsultorios;**

La respuesta del sistema presentará el listado de los consultorios incluidos en la tabla “consultorios”.

```
mysql> use citas;
Database changed
mysql> call listaconsultorios();
+-----+-----+
| ConNumero | ConNombre |
+-----+-----+
| 1 | Laboratorio Dental |
| 2 | Consultorio 1 |
| 3 | Toma de muestras |
+-----+-----+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)

mysql>
```

Ahora, para construir un procedimiento que liste el documento de identidad, nombres, apellidos y fecha de nacimiento de los pacientes, registrados en la base de Datos se utilizarian las siguientes sentencias:

Use citas	Se informa la base de datos en la que se va a trabajar
create procedure listapacientes()	Se informa que se va a crear el procedimiento, Llamado listapacientes
select pacIdentificacion, PacNombres, PacApellidos, PacFechaNacimiento from pacientes;	Instrucción SQL, para la consulta

A continuación se invoca el procedure y el resultado visualizado es:

```
call listapacientes();
```

La respuesta del sistema:

```
mysql>
mysql> call listaconsultorios();

+-----+-----+
| pacIdentificacion | PacNombres | PacApellidos | PacFechaNacimiento |
+-----+-----+
| 1098765678       | Carolina   | Rojas Zabala | 1984-06-28          |
| 37821200         | Evelia     | Arias Mendoza | 1970-03-25          |
| 37821203         | Mari Fernanda | Rodriguez Perez | 1970-07-28          |
| 63502720         | Maria Carolina | Rojas Perez | 1980-04-12          |
| 63502730         | Maria Alejandra | Torres Cañas | 1972-10-15          |
| 77191950         | Carlos Jose   | Arias Rojas | 1980-04-12          |
| 77191957         | Carlos Jose   | Arias Rojas | 1980-04-12          |
+-----+-----+

7 rows in set (0.00 sec)

Query OK, 0 rows affected (0.07 sec)
```

Para generar el listado de los pacientes ordenados alfabéticamente por apellidos, La sintaxis para el procedure sería:

```
Use citas Se informa la base de datos en la que se va a trabajar
create procedure ordenamientoapellidos() Se informa que se va a crear el procedimiento, Llamado ordenamientoapellidos
select pacIdentificacion, PacNombres, PacApellidos, PacFechaNacimiento from pacientes Instrucción SQL,
Orderby pacApellidos;
```

```
mysql> call ordenamientoapellidos();
+-----+-----+-----+-----+
| pacIdentificacion | PacNombres | PacApellidos | PacFechaNacimiento |
+-----+-----+-----+-----+
| 37821200          | Evelia     | Arias Mendoza | 1970-03-25        |
| 77191950          | Carlos Jose | Arias Rojas   | 1980-04-12        |
| 77191957          | Carlos Jose | Arias Rojas   | 1980-04-12        |
| 37821203          | Mari Fernanda | Rodriguez Perez | 1970-07-28        |
| 63502720          | Carolina    | Rojas Perez   | 1980-04-12        |
| 1098765678        | Maria Carolina | Rojas Zabala | 1984-06-28        |
| 63502730          | María Alejandra | Torres Cañas | 1972-10-15        |
+-----+-----+-----+-----+
7 rows in set (0.04 sec)
```

Toda instrucción SQL que se construya, puede ser almacenada en un procedure.

### 1.2. Procedimientos almacenados con parámetros

El parámetro es fundamental, al momento de utilizar criterios de selección en la cláusula where.

En los ejemplos anteriores después del nombre del procedure se incluían unos paréntesis, los cuales estaban vacíos; para estos ejemplos los paréntesis llevan información, y esta información se convertirá en el parámetro del procedimiento.

Para este proceso, iniciemos con listar los datos de todas las pacientes mujeres que están registradas en la entidad pacientes, para este caso el criterio de búsqueda es el campo sexo, que para este ejemplo se llama PacSexo y como la condición que es que sea mujer, este campo cuando se construyó la base de Datos, se declaró como tipo enum, con opciones M y F. Es importante aclarar que esta condición es sensible a mayúsculas y minúsculas

Según lo trabajado hasta el momento, la consulta SQL sería:

```
Select * from pacientes  
Where PacSexo='F ';
```

Es así que al aplicar esta instrucción a un procedure, la condición se convierte en un parámetro.

### 1.2.1. Creación de procedimientos Almacenados CON parámetros en MySQL

Tomando como referencia la instrucción construida en SQL anteriormente, apliquemos este concepto para construir el procedure en MySQL; al momento de construir el procedure se debe tener claro lo siguiente:

```
Create procedure <nombreprocedimiento> ( <nombreprámetro> <tipo_<br/>de _ dato _ del _ parámetro>)
```

Donde, <nombreprámetro> será el nombre que se va a utilizar como variable al momento de construir el where y tipo\_de\_dato\_del\_parámetro, será EXACTAMENTE EL MISMO TIPO DE DATO usado en el campo con el que se establece la condición.

En ese orden de ideas, como el campo que se tiene para la consulta es PacSexo y este campo solo recibe F o M, el tipo\_de\_dato\_del\_parámetro, es de tipo varchar(1). La sentencia para construir el procedure de este ejemplo es:

Use citas	Se informa la base de datos en la que se va a trabajar
create procedure listamujeres(vsexvarchar(1))	Vsex, nombre de a variable a utilizar y, varchar(1), donde vsex es la variable utilizada al crear el procedure
SELECT*FROM pacientes WHERE PacSexo=vsex;	Instrucción SQL, para la consulta, NOTE que PacSexo=a vsex, donde vsex es la variable utilizada al crear el procedure

Para ejecutar el procedure, se utiliza la misma palabra reservada call nombre\_del\_procedimiento y entre paréntesis el argumento a buscar, para este ejemplo la letra F, porque necesitamos listar los pacientes de sexo femenino. La sintaxis quedaría:

call listamujeres('F');	Note que dentro del paréntesis va la F, de femenino, y entre comilla sencilla, por ser un campo tipo varchar
-------------------------	--

Ahora visualicemos los nombres, apellidos y documento de identidad de los pacientes que se han realizado como tratamiento, un blanqueamiento dental. Recordemos que los datos básicos de los pacientes se encuentran

en la entidad “pacientes”, mientras que el tratamiento se encuentra en la entidad “tratamientos”, la consulta SQL sería de la siguiente manera:

```
select pac.PacIdentificacion, pac.PacNombres, pac.PacApellidos,
tra.TraDescripcion
from pacientes pac, tratamientos tra
where pac.pacIdentificacion = tra.TraPaciente and tra.
TraDescripcion='Blanqueamiento Dental';
```

Ahora incluyamos este instrucción SQL en un procedure.

Use citas	Se informa la base de datos en la que se va a trabajar
	create procedure listarblanqueamientos(tipostext)
	tipo, nombre de la variable a utilizar y, text, Porque el campo TraDescripcion, en la tabla es de ese tipo, va sin ancho, tal cual está en la tabla
	select pac.PacIdentificacion, pac.Pacnombres, pac.PacApellidos, traTraDescripcion from pacientes as pac inner join tratamientos as tra on pac.pacIdentificacion=tra.TraPaciente Where pac.pacIdentificacion=tra.TraPaciente and tra.TraDescripcion=tipo;
	Instrucción SQL, para la consulta, NOTE que TraDescripcion=tipo, donde tipo es la variable utilizada al crear el procedure

La llamada al procedure,

```
call listarblanqueamientos('Blanqueamiento Dental');
```

Note que dentro del paréntesis va Blanqueamiento Dental entre comilla sencilla, por ser un campo tipo text y por ser el tipo de tratamiento que deseamos listar.

La respuesta del sistema

```
mysql> call listarblanqueamientos<'Blanqueamiento Dental'>;
+-----+-----+-----+-----+
| pacIdentificacion | PacNombres | PacApellidos | TraDescripcion |
+-----+-----+-----+-----+
| 77191957          | Carlos Jose | Arias Rojas   | Blanqueamiento Dental |
| 37821203          | Mari Fernanda | Rodriguez Perez | Blanqueamiento Dental |
| 37821203          | Mari Fernanda | Rodriguez Perez | Blanqueamiento Dental |
+-----+-----+-----+-----+
3 rows in set (0.01 sec)

Query OK, 0 rows affected (0.05 sec)
```

También se pueden construir procedimientos almacenados para sentencias de inserción, modificación o eliminación de un registro a la tabla, la instrucción SQL para incluir un registro a la tabla “médicos” sería:

```
insert into medicos(MedIdentificacion, MedNombres, MedApellidos)
values ('63456789', 'Ramón', 'Arenas');
```

La codificación del procedimiento sería de la siguiente manera:

Use citas Se informa la base de datos en la que se va a trabajar

```
create procedure insertarmedico (identificacion char(10), nombre varchar(50), apellidos varchar(50))
```

Los argumentos del procedimiento corresponden a los campos de la table, los tipos deben ser iguales a los tipos de datos en la table. OJO deben llamarse diferente a los campos de la tabla

```
insert into medicos(MedIdentificacion, MedNombres, MedApellidos)
values (identificacion, nombre, apellidos);
```

Instrucción SQL, NOTE que en los values se referencian los argumentos del procedimiento

Para insertar un registro mediante el procedimiento insertar medico, se realiza la siguiente instrucción:

```
call insertarmedico('234567', 'Carlos', 'Pedraza');
```

Nombre de procedimiento | Son los datos a incluir en la tabla

### 1.2.2. Creación de procedimientos Almacenados CON parámetros tipo IN, en Oracle

En Oracle, TODOS los procedimientos almacenados, para manejo de bases de Datos, se manejan con argumentos, la sintaxis para la construcción o modificación de un procedimiento es:

```
CREATE [OR REPLACE]
PROCEDURE <procedure_name> [(<param1> [IN|OUT|IN OUT] <type>,
                                <param2> [IN|OUT|IN OUT] <type>, ...)]
IS -- Declaracion de variables locales
BEGIN
    -- Sentencias
[EXCEPTION]
    -- Sentencias control de excepcion
END [<procedure_name>];
```

El uso de OR REPLACE permite sobreescribir un procedimiento existente. Si se omite, y el procedimiento existe, el sistema producirá, un error.

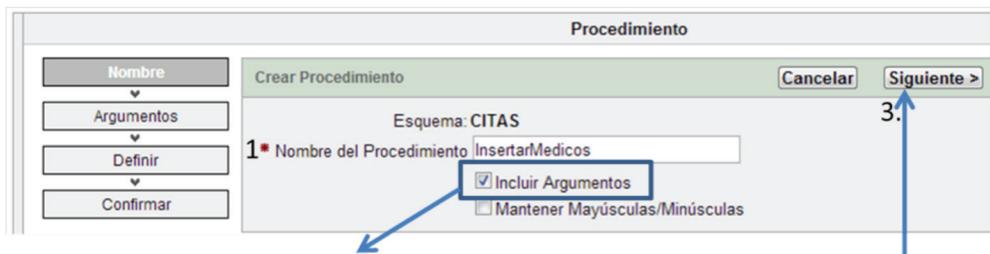
Inicialmente se va a realizar un procedimiento almacenado con argumentos tipo IN(entrada) para “ingresar” datos a la tabla “médicos”. Recuerde que la instrucción SQL para incluir registros a la tabla médicos es:

```
insert into medicos(MedIdentificacion, MedNombres, MedApellidos)  
values ( '1', 'nombre', 'apellidos');
```

En el entorno de trabajo de Oracle10g, se pueden codificar los procedimientos almacenados, haciendo clic en la icono Explorador de Objetos, opción crear, sub-opción Procedimiento.



Al hacer clic en la opción procedimiento, se visualiza la siguiente pantalla, para que se asigne el nombre del procedimiento y se continúe con el asistente, haciendo clic en el ícono siguiente:



2. Es muy importante activar esta opción para el manejo de los Argumentos (in, out o in out) del procedimiento

3. Despues de asignar el nombre, y activar la opción “incluir argumentos, se hace clic en Siguiente”

El siguiente paso en el asistente es configurar los argumentos del procedimiento, el número de argumentos depende de la cantidad de campos a manipular en la tabla, en este ejemplo se tienen tres (3) argumentos, porque la tabla médicos tiene tres campos, así

Nombre	Tipo	Ancho
MedIdentificacion	Char	10
MedNombres	VARCHAR	50
MedApellidos	VARCHAR	50

A continuación, la presentación del sistema:

Note que los argumentos del procedimiento corresponden a los campos de la tabla y manejan el mismo tipo de dato del campo, y por ser datos que se digitán por el usuario, son de tipo entrada (IN)

Identifique los argumentos que desea incluir en el procedimiento. Los argumentos son parámetros que se transfieren o devuelven de los procedimientos. Si desea que el argumento sea nulo, escriba Nulo como valor por defecto.

Una vez configurados los parámetros del procedimiento, se hace clic en el icono de “siguiente”; el asistente nos presenta la pantalla para que se digite la respectiva instrucción SQL, en este caso con la cláusula insert

**Procedimiento**

**Crear Procedimiento**

Cancelar < Anterior Siguiente >

Esquema: CITAS  
Nombre del Procedimiento: INSERTARMEDICOS

\* Cuerpo del Procedimiento:

```
insert into medicos(MedIdentificacion, MedNombres, MedApellidos)
values ( IdentificacionMed, NombreMed, ApellidoMed);
```

1. Los campos deben llamarse igual y estar en el mismo orden que en la tabla.  
2. Estos nombres corresponden a los argumentos creados en el paso anterior y deben estar en el mismo orden que en el insert

Utilice esta página para introducir el bloque PL/SQL que desee utilizar como cuerpo del procedimiento. El cuerpo del procedimiento es todo lo incluido entre BEGIN y END;  
Por ejemplo, si define un parámetro IN del tipo VARCHAR2 denominado p\_name en el paso anterior, podría introducir lo siguiente para el cuerpo del procedimiento:  
`http.p('Hola '||p_name);`

Después de digitar la respectiva instrucción SQL y al hacer clic en siguiente, el sistema nos presenta la información del procedimiento que se acabó de construir para que se dé por terminada la construcción del procedimiento, haciendo clic en el botón terminar. Si es necesario realizar alguna corrección entonces se hace clic en el botón anterior.

**Procedimiento**

**Crear Procedimiento**

Cancelar < Anterior Terminar

Esquema: CITAS  
Tipo de Objeto: Procedimiento  
Objeto: INSERTARMEDICOS

SQL O SQL

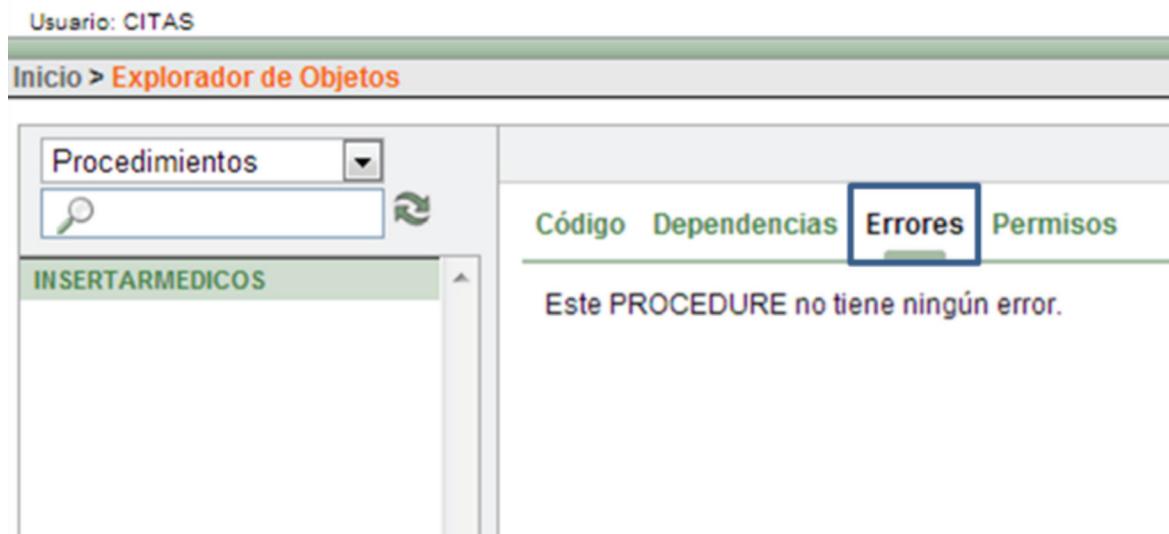
Por defecto aparece este icono, debo hacer clic en él, para visualizar el código SQL, automáticamente el icono cambia

```
create or replace procedure "INSERTARMEDICOS"
(IdentificacionMed IN CHAR,
NombreMed IN VARCHAR2,
ApellidoMed IN VARCHAR2)
is
begin
insert into medicos(MedIdentificacion, MedNombres, MedApellidos)
values ( IdentificacionMed, NombreMed, ApellidoMed);

end;
/
```

Al hacer clic en terminar, automáticamente aparece el procedimiento creado en el panel izquierdo del explorador de objetos, por seguridad se debe verificar si se tiene errores o no; esto se realiza con el ícono errores. Cuando el procedimiento tiene errores aparece con una franja vertical roja antes del nombre de dicho procedimiento.

## ORACLE® Database Express Edition



### 1.2.3. Creación de procedimientos Almacenados CON parámetros tipo OUT, en Oracle

Los procedimientos almacenados con argumentos tipo OUT, son los utilizados para generar las salidas del sistema, esto significa que se han trabajado las instrucciones con la cláusula select. La cláusula select puede ser utilizada de manera simple, es decir sin condiciones, o de manera compuesta, con condiciones mediante la utilización del where. El manejo del select, se ha trabajado en el objeto de aprendizaje anterior denominado “**Construir sentencias SQL para la definición y manipulación del modelo de base de datos**”, y también en los ejercicios propuestos en este objeto de aprendizaje, manejando MySQL; retomemos el ejemplo del listado de los consultorios que se realizó en MySQL y trabajémoslo desde ORACLE.

Oracle para la presentación de datos, mediante la cláusula select, utiliza el argumento OUT asociado al tipo de datos especial denominado “cursor”, notará que en la instrucción se trabaja una nueva palabra reservada

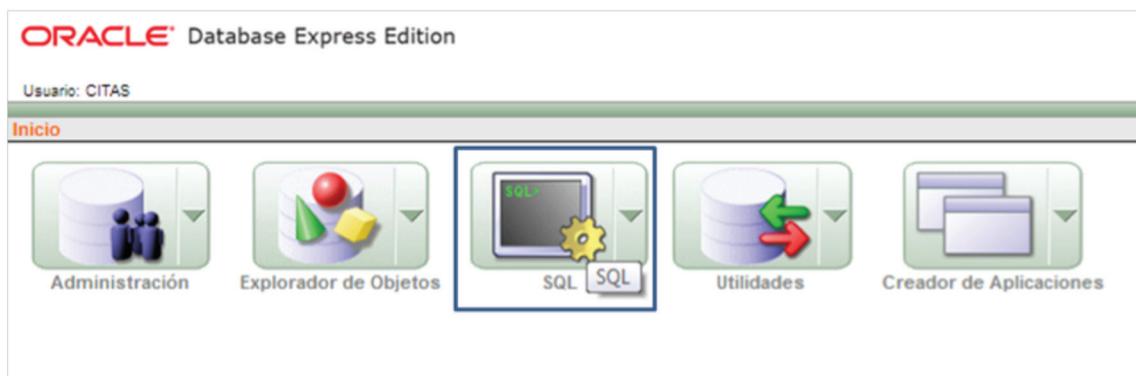
"SYS\_REFCURSOR", este tipo de dato es el que nos permitirá generar los datos que trae la consulta.

En este momento se va a listar los consultorios

Nombre	Tipo	Ancho
ConNumero	Integer	3
ConNombre	Varchar	50

Los pasos a seguir en el asistente para la construcción de procedimientos almacenados en Oracle es:

- a. clicen la icono Explorador de Objetos, opción SQL, donde aparecerá el espacio para digitar la instrucción SQL correspondiente a la creación del procedimiento:



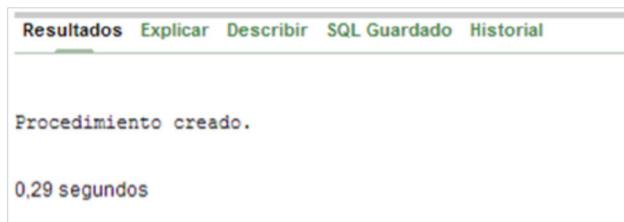
- b. Codificación del procedure:

```

ORACLE Database Express Edition
Usuario: CITAS
Inicio > SQL > Comandos SQL
Guardado: 2013-05-14 10:30:00
Ejecutar: 2013-05-14 10:30:00
? Desconectar Ayuda
1. Nombre del argumento
2. Tipo "out" (salida)
3. Tipo de argumento: SYS_REFCURSOR
create or replace PROCEDURE proconsultorios
as
BEGIN
OPEN datosconsul FOR
SELECT *
FROM consultorios;
END proconsultorios;
  
```

1. IMPORTANTE: Al final de la consulta y del procedure debe estar el respectivo punto y coma

- c. Al final clic en el botón ejecutar, para que el procedure quede grabado en el sistema.



#### **1.2.4. Creación de procedimientos Almacenados CON parámetros tipo IN y OUT, en Oracle**

Listar los pacientes de tipo Femenino.

```

ORACLE Database Express Edition
Usuario: CITAS
Inicio > SQL > Comandos SQL
Guardado Ejecutar
create or replace PROCEDURE proc_mujeres ( sex IN varchar, cur_paci out SYS_REFCURSOR)
as
BEGIN
OPEN cur_paci FOR
SELECT *
FROM pacientes
WHERE PacSexo= sex;
END proc_mujeres;
  
```

1. Nombre del procedimiento, éste se debe finalizar con un end seguido de punto y coma

2. Nombre del argumento; se crea para digitar el sexo, DEBE utilizarse en el where para comprar el campo de la tabla con la variable, la condición termina con un punto y coma

3. Tipo "IN" (Entrada), para digitar el sexo

4. Tipo del dato de entrada, el mismo de la tabla

5. Nombre del cursor creado para el resultado de la consulta

6. Tipo "out" por ser el del cursor

7. Tipo de argumento: SYS\_REFCURSOR

Listar los datos de los pacientes que se han realizado como tratamiento un "Blanqueamiento Dental"

ORACLE Database Express Edition

Usuario: CITAS

Inicio > SQL > Comandos SQL

Confirmación Automática Mostrar 10 ▾

create or replace PROCEDURE proc\_tratamiento(1 trata IN varchar, 2 cur\_trata OUT SYS\_REFCURSOR) as BEGIN OPEN cur\_trata FOR select pac.PacIdentificacion, pac.PacNombres, pac.PacApellidos, tra.TraDescripcion from pacientes pac, tratamientos tra where pac.pacIdentificacion = tra.TraPaciente and tra.TraDescripcion=trata; 2 END proc\_tratamiento; 1

Guardar Ejecutar

1. Nombre del procedimiento, éste se debe finalizar con un end seguido de punto y coma  
 2. Nombre del argumento; se crea para digitar el nombre del tratamiento, DEBE utilizarse en el where para comprar el campo de la tabla con la variable, la condición termina con un punto y coma  
 3. Nombre del cursor creado para el resultado de la consulta

## 2. FUNCIONES

### 2.1. FUNCIONES EN MYSQL

MySQL, para crear funciones ofrece la directiva CREATE FUNCTION, una función se diferencia de un procedimiento, porque devuelve valores.

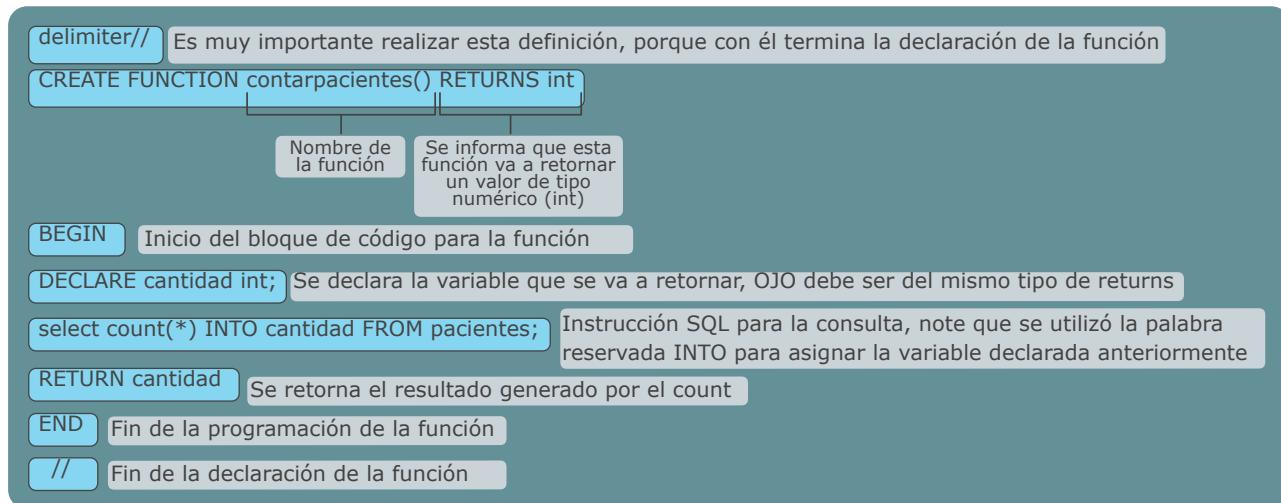
Estos valores pueden ser utilizados como argumentos para instrucciones SQL, un ejemplo podría ser las funciones MAX() o COUNT().

Debido a que en este proceso se devuelven valores, es obligatorio utilizar la cláusula RETURNS, al momento de definir una función y sirve para especificar el tipo de dato que será devuelto (sólo el tipo de dato, no el dato).

Sintaxis:

```
delimiter//  
CREATE FUNCTION nombre_funcion (parámetro)  
Returns tipo  
BEGIN  
[características] definición  
END  
//
```

Como ejemplo, se va a construir una función para que cuente los pacientes que están registrados en la tabla pacientes, la sintaxis sería la siguiente:



## 2.2. Funciones en Oracle

Una función en Oracle, al igual que en MYSQL, de vuelve valores, esa es la diferencia respecto a un procedimiento almacenado.

En esta función se va a contar cuantos pacientes se tienen registrados en la tabla “pacientes”,

ORACLE Database Express Edition

Usuario: CITAS

Inicio > SQL > Comandos SQL

create or replace FUNCTION Contrarpacientes RETURN NUMBER IS  
cantidad NUMBER;  
BEGIN  
SELECT count(\*) INTO cantidad FROM pacientes;  
RETURN cantidad;  
END;

Guardar Ejecutar

1. Nombre de la función, éste se debe finalizar con un end seguido de punto y coma  
2. Todas función retorna un valor, SIEMPRE VA en la construcción de la función.  
3. Tipo de dato a retornar, normalmente es NUMBER  
4. Declaración de la variable que va a recibir el valor de la consulta. Esta variable DEBE ser del MISMO tipo del return(3) y se debe retornar después de la consulta.

Al finalizar la codificación se hace clic en el botón ejecutar, automáticamente el sistema reporta la creación de la función.

Resultados	Explicar	Describir	SQL Guardado	Historial
Función creada.				

0,00 segundos

### 3. TRIGGERS

Los triggers, también conocidos como disparadores o desencadenadores, son subrutinas que se ejecutan de manera automática, cuando sucede algún evento sobre las tablas asociadas a la base de datos. Los triggers, normalmente son utilizados para realizar auditorías a la base de datos

Los eventos que activan un trigger pueden ser las sentencias INSERT, DELETE, UPDATE que pueden modificar los datos de una tabla. Los triggers se pueden ejecutar antes (BEFORE) y/o después (AFTER) de que sean modificados los datos.

Los triggers tienen dos palabras clave, OLD y NEW que se refieren a los valores que tienen las columnas antes y después de la modificación. Los INSERT permiten NEW, los DELETE sólo OLD y los UPDATE ambas.

#### 3.1. CREACION DE UN TRIGGER EN MySQL

Se puede definir un disparador para que se active antes de borrar un registro o después que el registro sea actualizado.

Sintaxis:

```
CREATE TRIGGER <NombreTrigger>
BEFORE AFTER
INSERT UPDATE DELETE
ON
```

```
<nombredelatabla>
FOR EACH ROW
BEGIN
<sentencias SQL>
END;
```

Para trabajar el tema de disparadores desde MySQL, se va a realizar auditoria en la manipulación de los datos para la tabla “pacientes”, esta tabla tiene la siguiente estructura:

Column Name	Datatype	NOT NULL	AUTO INC
PacIdentificacion	CHAR(10)	✓	
PacNombres	VARCHAR(50)	✓	
PacApellidos	VARCHAR(50)	✓	
PacFechaNacimiento	DATE	✓	
PacSexo	ENUM('M','F')	✓	

El control se va a llevar específicamente en la actualización de los datos (comando update) y en la eliminación de los registros (comando delete); los pasos a seguir son los siguientes:

Paso 1. Crear la tabla auditoria\_pacientes: En esta tabla se deben incluir los campos que se tenían inicialmente, los campos que fueron modificados, la fecha de modificación, el usuario que realizó la acción (update o delete), y la acción que se realizó, es así que la tabla quedaría con los siguientes campos.

Table Name: auditoria\_pacientes Database: citas

Columns and Indices Table Options Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC
id_audi	INTEGER	✓	✓
audi_NombreAnterior	VARCHAR(50)		
audi_ApellidAnterior	VARCHAR(50)		
audi_FechaAnterior	DATE		
audi_SexoAnterior	ENUM('M','F')		
audi_NombreNuevo	VARCHAR(50)		
audi_ApellidoNuevo	VARCHAR(50)		
audi_FechaNueva	DATE		
audi_SexoNuevo	ENUM('M','F')		
audi_FechaModificacion	DATETIME		
audi_usuario	VARCHAR(45)		
audi_PadIdentificacion	CHAR(10)	✓	
audi_accion	VARCHAR(45)	✓	

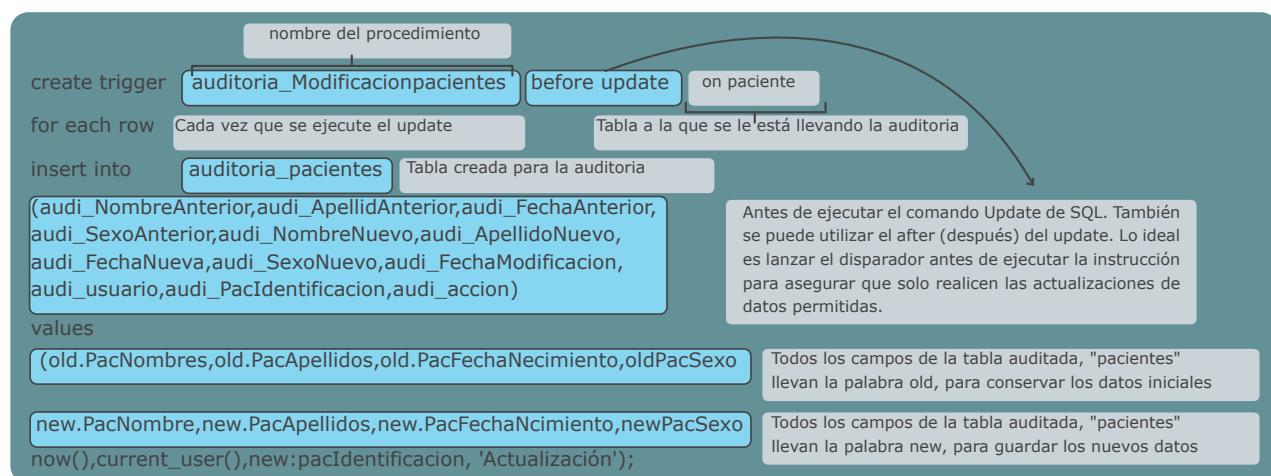
Es muy importante manejar el orden de los campos y los tipos de los datos, porque de lo contrario se generaran muchos errores de sintaxis, preste atención a los siguientes detalles

Table Name: auditoria_pacientes		Database: citas	
Columns and Indices		Table Options	
Column Name	Datatype	NOT NULL	AUTO INC
id_audi	INTEGER	✓	✓
audi_NombreAnterior	VARCHAR(50)	PacNombres	VARCHAR(50)
audi_ApellidAnterior	VARCHAR(50)	PacApellidos	VARCHAR(50)
audi_FechaAnterior	DATE	PacFechaNacimiento	DATE
audi_SexoAnterior	ENUM('M','F')	PacSexo	ENUM('M','F')
audi_NombreNuevo	VARCHAR(50)	PacNombres	VARCHAR(50)
audi_ApellidoNuevo	VARCHAR(50)	PacApellidos	VARCHAR(50)
audi_FechaNueva	DATE	PacFechaNacimiento	DATE
audi_SexoNuevo	ENUM('M','F')	PacSexo	ENUM('M','F')
audi_FechaModificacion	DATETIME	now()	
audi_usuario	VARCHAR(45)	current_user()	
audi_PacIdentificacion	CHAR(10)	✓	
audi_accion	VARCHAR(45)	✓	

- a. Al momento de realizar una actualización de datos, mediante la instrucción **update**, estos campos guardarán la información que se tenía inicialmente. Note que están en estricto orden respecto a la tabla asociada a la auditoria y se manejan los mismos tipos de datos.
- b. Al momento de realizar una actualización de datos, mediante la instrucción **update**, estos campos guardarán la información que se modificó al utilizar el **update**. Note que están en estricto orden respecto a la tabla asociada a la auditoria y se manejan los mismos tipos de datos. Si lo que se realizan es el borrado de datos mediante la instrucción **delete** estos campos estarán vacíos y aparecerán con la palabra null.
- c. La información de estos campos se toma directamente del sistema mediante funciones ya predefinidas como se puede observar en el gráfico.
- d. Aunque es la llave principal (primary key) en la tabla “pacientes”, se coloca al final de todos los campos que se van a auditar, porque NUNCA se modifica un primary key, y es el último argumento al momento de trabajar el código SQL para el update y el delete.

Paso 2. Crear los triggers para almacenar los cambios generados a la tabla pacientes, específicamente registros actualizados y eliminados.

Paso 2.1 Trigger para auditar registros actualizados: El objetivo de este trigger es llevar el control de los cambios realizados a los datos de los pacientes ya incluidos en la tabla “pacientes”. La instrucción es la siguiente:

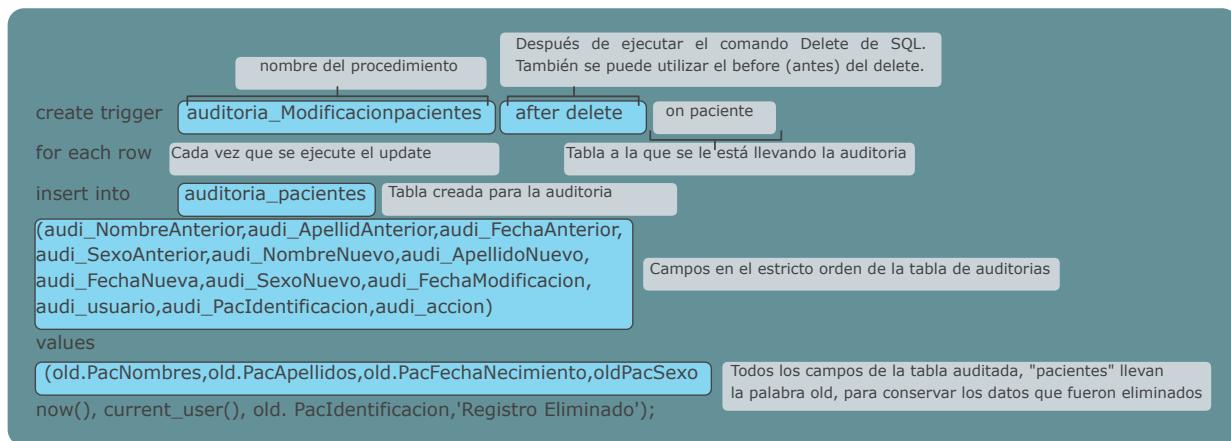


Ahora a verificar la función del trigger, como se configuró con la opción de actualización, se va a modificar un registro con la instrucción update, la sintaxis podría ser:

```

update pacientes set PacNombres='Josexx', PacApellidos='Rozo
Torres', PacFechaNacimiento='1980-04-04', PacSexo='M'
where PacIdentificacion='77191950';
  
```

Paso 2.2 Trigger para auditar registros eliminados: Una vez registrado un paciente, es muy importante llevar el histórico de estos, es por ello, que si se requiere eliminar un registro, se deben dejar almacenados en la tabla de históricos, los datos que se eliminaron con el respectivo usuario y la fecha en la que se realizó el proceso; esa es la función de la tabla de auditorias.



## VALUES EN LOS TRIGGERS

Note que los values en El disparador, "Modifica\_auditoria\_pacientes", maneja las palabras reservadas "old" y "new"; porque en este proceso es importante dejar la evidencia de los valores iniciales y los valores anteriores.

Los campos que contengan old.nombre\_campo, son los encargados de conservar los datos a modificar, y los que contengan new.nombre\_campo, son los que tomaran los nuevos datos.

Por ejemplo, Un paciente informa que su nombre está mal escrito y solicita que sea cambiado su nombre "C@rlos Eurelio", por el correcto, "Carlos Aurelio". El campo que se manejaría con el old, por ser el dato inicial, seria "C@rlos Eurelio", y el que se manejaría con el new seria "Carlos Aurelio".

También puede identificar que en el trigger "auditoria\_pacientesEliminados", solo se maneja la palabra reservada old, porque en este disparador no se modifica ningún dato, solo se eliminan registros.

## 3.2. BORRADO DE UN TRIGGER EN MySQL

Puede suceder que por alguna situación se necesita borrar un trigger, la sintaxis para eliminarlos es:

```
drop trigger <Nombre_del_trigger>
```

### 3.3. CREACION DE UN TRIGGER EN Oracle

Para crear un disparador en Oracle, se debe crear, al igual que en MYSQL, una tabla para manejar el histórico de las transacciones realizadas, la tabla que se va a crear es la siguiente y los campos deben manejar los tipos de datos acorde con la tabla a auditar:

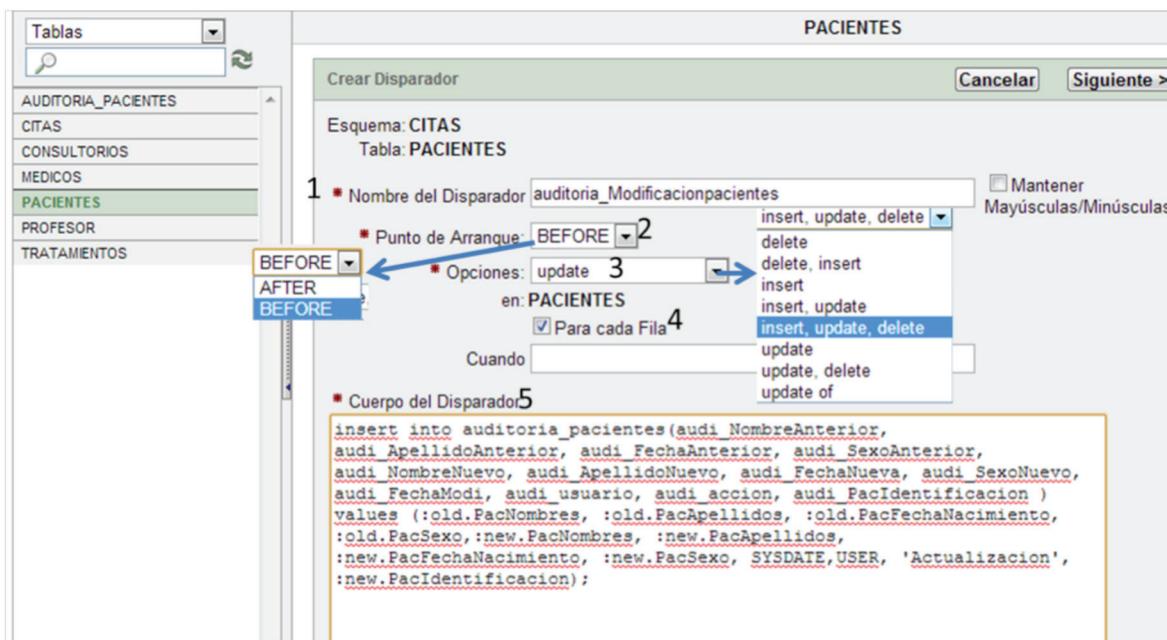
AUDITORIA_PACIENTES					
Tabla	Datos	Índices	Modelo	Restricciones	Permisos
<a href="#">Agregar Columna</a> <a href="#">Modificar Columna</a> <a href="#">Cambiar Nombre de Columna</a> <a href="#">Borrar Columna</a> <a href="#">Cam</a>					
Nombre De Columna	Tipo De Dato	Nulo	Valor Por Defecto	Clave Primaria	
ID_AUDI	NUMBER	No	-	1	
AUDI_NOMBREANTERIOR	VARCHAR2(50)	Yes	-	-	
AUDI_APELLIDOANTERIOR	VARCHAR2(50)	Yes	-	-	
AUDI_FECHAANTERIOR	DATE	Yes	-	-	
AUDI_SEXOANTERIOR	VARCHAR2(1)	Yes	-	-	
AUDI_NOMBRENUEVO	VARCHAR2(50)	Yes	-	-	
AUDI_APELLIDONUEVO	VARCHAR2(50)	Yes	-	-	
AUDI_FECHANUEVA	DATE	Yes	-	-	
AUDI_SEXONUEVO	VARCHAR2(1)	Yes	-	-	
AUDI_FECHAMODI	DATE	Yes	-	-	
AUDI_USUARIO	VARCHAR2(45)	Yes	-	-	
AUDI_ACCION	VARCHAR2(45)	Yes	-	-	
AUDI_PACIDENTIFICACION	CHAR(10)	Yes	-	-	
1 - 13					

PACIENTES					
Tabla	Datos	Índices	Modelo	Restricciones	Permisos
<a href="#">Agregar Columna</a> <a href="#">Modificar Columna</a> <a href="#">Cambiar Nombre de Columna</a> <a href="#">Borrar Columna</a> <a href="#">Cam</a>					
Nombre De Columna	Tipo De Dato	Nulo	Valor Por Defecto	Clave Primaria	
PACIDENTIFICACION	CHAR(10)	No	-	1	
PACNOMBRES	VARCHAR2(50)	Yes	-	-	
PACAPELLIDOS	VARCHAR2(50)	Yes	-	-	
PACFECHANACIMIENTO	DATE	Yes	-	-	
PACSEXO	CHAR(1)	Yes	-	-	
1 - 5					

Luego de construir la tabla a auditar, y la del histórico de la auditoria, se procede a configurar el disparador, para este proceso, se ubica sobre el nombre de la tabla a auditar, para este caso (pacientes), y se busca sobre las opciones de la tabla, el icono “disparadores”



Luego se hace clic en el botón crear, a continuación cinco aspectos a tener en cuenta para codificar el disparador.

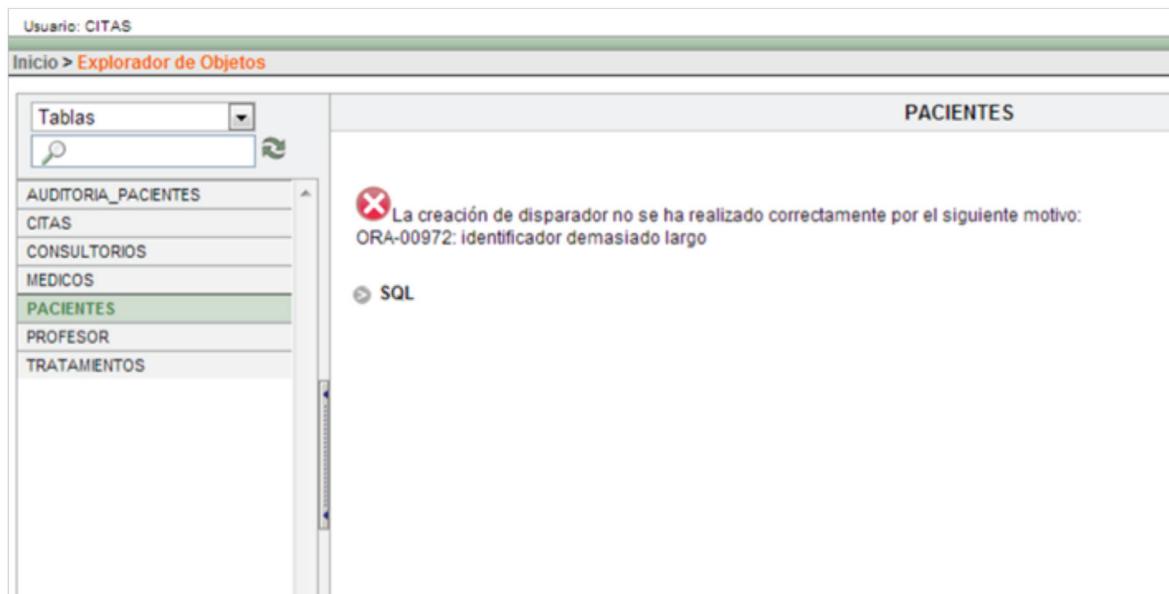


1. El nombre normalmente se convierte a mayúsculas, excepto si se activa la opción de Mantener Mayúsculas/Minúsculas.
2. El punto de arranque de un disparador, es Antes o después de ejecutar una instrucción SQL
3. Las opciones corresponden a la instrucción SQL a ejecutar, como se puede observar puede Oracle permite todas las cláusulas SQL para manipulación de las tablas.
4. Es muy importante que esté activa la opción “para cada fila”, porque

esto nos permite, que cada vez que se ejecute la instrucción SQL, se active el disparador.

5. El código utilizado en el cuerpo del disparador debe contener la instrucción para insertar el respectivo registro en la tabla creada para la auditoria, para nuestro ejemplo “auditoria\_pacientes”, con los campos en el respectivo orden para guardar el histórico. Recuerde que existen las palabras reservadas old(información actual, noté que antes de la palabra old van dos puntos) y new (para la nueva captura, igual antes de la palabra new, van los dos puntos).

El sistema nos presenta el código construido, para analizar el código construido, se puede hacer clic en terminar o en anterior, en ocasiones el código SQL puede estar bien, pero pueden suceder errores como el siguiente:



Es así como se debe volver a realizar el proceso, teniendo la precaución de manejar nombres menos largos.

Inicio > Explorador de Objetos

**PACIENTES**

Crear Disparador

Esquema: CITAS  
Tabla: PACIENTES

\* Nombre del Disparador: auditModificaPac  Mantener Mayúsculas/Minúsculas

\* Punto de Arranque: BEFORE

\* Opciones: update   
en: PACIENTES  
 Para cada Fila  
Cuando:

\* Cuerpo del Disparador:

```
insert into auditoria_pacientes(audi_NombreAnterior,
audi_ApellidoAnterior, audi_FechaAnterior, audi_SexoAnterior,
audi_NombreNuevo, audi_ApellidoNuevo, audi_FechaNueva, audi_SexoNuevo,
audi_FechaModi, audi_usuario, audi_accion, audi_FacIdentificacion )
values (:old.PacNombres, :old.PacApellidos, :old.PacFechaNacimiento,
:old.PacSexo,:new.PacNombres, :new.PacApellidos,
:new.PacFechaNacimiento, :new.PacSexo, SYSDATE,USER, 'Actualizacion',
:new.PacIdentificacion);|
```

Clic en siguiente y luego terminar, automáticamente se visualiza la siguiente presentación para activar el respectivo disparador.

Inicio > Explorador de Objetos

**PACIENTES**

Tabla	Datos	índices	Modelo	Restricciones	Permisos	Estadísticas	Valores por l
AUDITORIA_PACIENTES							
CITAS							
CONSULTORIOS							
MEDICOS							
<b>PACIENTES</b>							
PROFESOR							
TRATAMIENTOS							

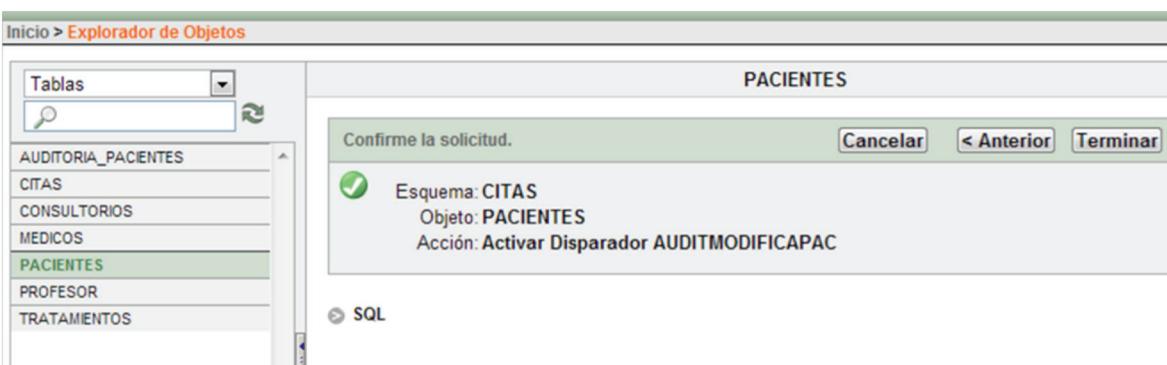
Borrar Crear Activar Desactivar

Nombre Del Disparador	Tipo De Disparador	Evento Disparador	Estado
AUDITMODIFICAPAC	BEFORE EACH ROW	UPDATE	ENABLED
1 - 1			

Se debe hacer clic en el boto activar, para que el asistente nos permita activar el respectivo disparador, probablemente aparezca más de un disparador para activar.



Una vez seleccionado el disparador, se hace clic en el botón siguiente



Y para terminar el proceso de activación del disparador se hace clic en el botón terminar, si se logró activar correctamente el disparador a la tabla aparecerá la siguiente presentación.

**ORACLE® Database Express Edition**

Usuario: CITAS

Inicio > Explorador de Objetos

<input style="width: 100%;" type="button" value="Disparadores"/> <input style="width: 100%; background-color: #0070C0; color: white; font-weight: bold;" type="button" value="Auditmodificapac"/> <input style="width: 100%;" type="button" value="BI_AUDITORIA_PACIENTES"/> <input style="width: 100%;" type="button" value="BI_CITAS"/> <input style="width: 100%;" type="button" value="BI_PROFESOR"/> <input style="width: 100%;" type="button" value="BI_TRATAMIENTOS"/>	<p style="text-align: center;"><b>AUDITMODIFICAPAC</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #0070C0; color: white;">Detalles de Objeto</th> <th>Código</th> <th>Errores</th> <th>SQL</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;"><input type="button" value="Borrar"/></td> <td style="text-align: center;"><input type="button" value="Compilar"/></td> <td style="text-align: center;"><input type="button" value="Descargar"/></td> <td style="text-align: center;"><input type="button" value="Desactivar"/></td> </tr> <tr> <td colspan="4"><b>Detalles</b></td> </tr> <tr> <td>PACIDENTIFICACION</td> <td>PACIENTES</td> <td>CITAS</td> <td>New</td> </tr> <tr> <td>PACSEXO</td> <td>PACIENTES</td> <td>CITAS</td> <td>-</td> </tr> <tr> <td>PACFECHANACIMIENTO</td> <td>PACIENTES</td> <td>CITAS</td> <td>-</td> </tr> <tr> <td>PACNOMBRES</td> <td>PACIENTES</td> <td>CITAS</td> <td>-</td> </tr> <tr> <td>PACAPELLIDOS</td> <td>PACIENTES</td> <td>CITAS</td> <td>-</td> </tr> <tr> <td colspan="4" style="text-align: right;">1 - 5</td> </tr> </tbody> </table>	Detalles de Objeto	Código	Errores	SQL	<input type="button" value="Borrar"/>	<input type="button" value="Compilar"/>	<input type="button" value="Descargar"/>	<input type="button" value="Desactivar"/>	<b>Detalles</b>				PACIDENTIFICACION	PACIENTES	CITAS	New	PACSEXO	PACIENTES	CITAS	-	PACFECHANACIMIENTO	PACIENTES	CITAS	-	PACNOMBRES	PACIENTES	CITAS	-	PACAPELLIDOS	PACIENTES	CITAS	-	1 - 5			
Detalles de Objeto	Código	Errores	SQL																																		
<input type="button" value="Borrar"/>	<input type="button" value="Compilar"/>	<input type="button" value="Descargar"/>	<input type="button" value="Desactivar"/>																																		
<b>Detalles</b>																																					
PACIDENTIFICACION	PACIENTES	CITAS	New																																		
PACSEXO	PACIENTES	CITAS	-																																		
PACFECHANACIMIENTO	PACIENTES	CITAS	-																																		
PACNOMBRES	PACIENTES	CITAS	-																																		
PACAPELLIDOS	PACIENTES	CITAS	-																																		
1 - 5																																					

Por seguridad es importante verificar el botón de errores, para estar completamente seguros que el proceso quedó correctamente.

**ORACLE® Database Express Edition**

Usuario: CITAS

Inicio > Explorador de Objetos

<input style="width: 100%;" type="button" value="Disparadores"/> <input style="width: 100%; background-color: #0070C0; color: white; font-weight: bold;" type="button" value="Auditmodificapac"/> <input style="width: 100%;" type="button" value="BI_AUDITORIA_PACIENTES"/> <input style="width: 100%;" type="button" value="BI_CITAS"/> <input style="width: 100%;" type="button" value="BI_PROFESOR"/> <input style="width: 100%;" type="button" value="BI_TRATAMIENTOS"/>	<p style="text-align: center;"><b>AUDITMODIFICAPAC</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #0070C0; color: white;">Detalles de Objeto</th> <th>Código</th> <th>Errores</th> <th>SQL</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Este disparador no tiene ningún error.</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> </tbody> </table>	Detalles de Objeto	Código	Errores	SQL	Este disparador no tiene ningún error.			
Detalles de Objeto	Código	Errores	SQL						
Este disparador no tiene ningún error.									

Ahora para verificar que el disparador está funcionando y que se están enviando los datos a la tabla auditoria\_pacientes; en el editor de comandos de SQL, al cual se llega haciendo clic en el icono SQL



Se va a digitar la siguiente instrucción SQL:

```
update pacientes set PacNombres='Josexx', PacApellidos='Rozo  
Torres', PacFechaNacimiento='', PacSexo='M' where  
PacIdentificacion='37821203'; Esta identificación debe ser de un  
registro que esté incluido en la tabla pacientes.
```

Y para verificar que la información que se acabó de modificar quedó en la tabla auditoria\_pacientes, se utiliza la siguiente instrucción:

```
select * from auditoria_pacientes;
```

### 3.4. BORRADO DE UN TRIGGER EN Oracle

Para eliminar un disparador, el procedimiento es bastante sencillo, solo es cuestión de ubicarse en el disparador, luego hacer clic en la opción "detalles de Objeto" y clic en borrar

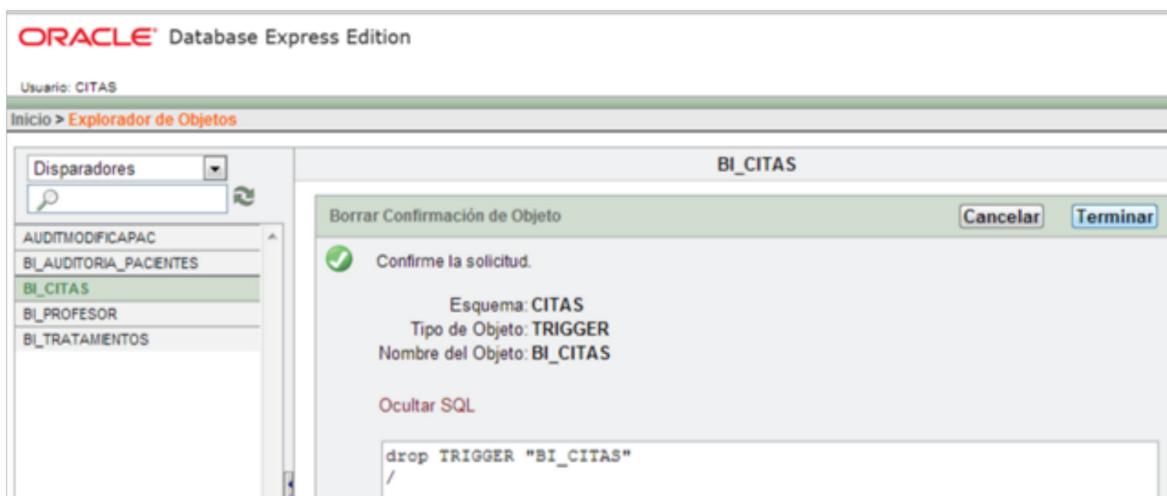
The screenshot shows the Oracle Database Express Edition interface. The left sidebar lists several triggers: AUDITMODIFICAPAC, BI\_AUDITORIA\_PACIENTES, BI\_CITAS, BI\_PROFESOR, and BI\_TRATAMIENTOS. The BI\_CITAS trigger is currently selected and highlighted with a green background. On the right, the 'BI\_CITAS' trigger details are displayed in a table:

Detalles de Objeto		Código	Errores	SQL
<input type="button" value="Borrar"/> <input type="button" value="Compilar"/> <input type="button" value="Descargar"/> <input type="button" value="Desactivar"/>				
Detalles				
Estado del Objeto	VALID			
Estado del Disparador	ENABLED			
Tipo de Disparador	BEFORE EACH ROW			
Evento Disparador	INSERT			
Tipo de Objeto Base	TABLE			
Propietario de Objeto Base	CITAS			
Nombre de Objeto Base	CITAS			
Nombre de Columna	-			
Nombres de Referencia	REFERENCING NEW AS NEW OLD AS OLD			
Cláusula When	-			
Descripción	"BI_CITAS" before insert on "CITAS" for each row			
Tipo de Acción	PL/SQL			

Below this, there is a section titled 'Columnas' (Columns) with the following table:

Columna	Tabla	Propietario De Tabla	Uso
CITNUMERO	<u>CITAS</u>	CITAS	New modified
			1 - 1

Automáticamente el sistema presenta el pantallazo para la confirmación de la eliminación del disparador



## GLOSARIO

---

BEGIN: Limitador del procedimiento

CALL: llamar un procedimiento previamente creado

DELIMITER: Restablece el punto y coma como delimitador.

END: Fin de nuestro procedimiento

IN : Nos indica que el parámetro del trigger será de entrada

INTOUT : Indica que el parámetro del trigger será tanto de Entrada como de salida

OUT : Nos indica que el parámetro del trigger será de salida

Trigger: Disparador

## BIBLIOGRAFÍA

---

Manual de Referencia MySQL, disponible en <http://dev.mysql.com/doc/refman/5.0/es/index.html>

Manual de Referencia MySQL, disponible en <http://dev.mysql.com/doc/refman/5.0/es/stored-procedures.html>

Manual de Referencia Oracle, disponible en: <http://www.oracle.com/technetwork/index.html>

Oracle Database 11g: Develop PL/SQL Program Units. Volumen I. Student guide.

D49986GC12, Edition 1.2. April 2009. D59428. ORACLE.

<b>Control de documento</b> <b>Construcción Objeto de Aprendizaje</b> <b>Lenguaje Transaccional en SQL</b>	
Desarrollador de contenido Experto temático	Magda Milena García Gamboa
Asesor pedagógico	Rafael Neftalí Lizcano Reyes
Producción Multimedia	Luis Fernando Botero Mendoza William Fernando Ramírez V.
Programadores	Daniel Eduardo Martínez
Líder expertos temáticos	Ana Yaqueline Chavarro Parra
Líder línea de producción	Santiago Lozada Garcés

