MBDA 01

AUTOESTUDIO 14/03/19

Línea horizontal

# AUTORES

Laura Izquierdo, Yarit Villalobos

# **INVESTIGACIÓN**

# **A. Acciones referenciales**

# ¿Para qué sirven las acciones referenciales?

Integridad de referencias. Para conservar la validez sobre los valores que se está referenciando. Es decir modificar una tabla que ha sido eliminada o actualizada sin que pierda validez.

Integridad referencial es direccional

# ¿Qué acciones soporta ORACLE? ¿Qué permite hacer cada una de ellas?

Las restricciones de integridad referencial en cascada permiten definir las acciones que Motor de base de datos lleva a cabo cuando un usuario intenta **eliminar** o **actualizar** una clave a la que apuntan las claves externas existentes. Se pueden definir las acciones en cascada.

**NO ACTION**

Motor de base de datos genera un error y se revierte la acción de eliminación o actualización de la fila de la tabla primaria.

**CASCADE**

Si esa fila se actualiza o elimina en la tabla primaria, las filas correspondientes se actualizan o eliminan en la tabla de referencia. CASCADE no se puede especificar si una columna timestamp es parte de una clave externa o de la clave con referencia. ON DELETE CASCADE no se puede especificar en una tabla que tenga un desencadenador INSTEAD OF DELETE. No se puede especificar ON UPDATE CASCADE para las tablas que tienen desencadenadores INSTEAD OF UPDATE.

**SET NULL**

Cuando se actualiza o elimina la fila correspondiente en la tabla primaria, todos los valores que componen la clave externa se establecen en NULL. Para que esta restricción se ejecute, las columnas de clave externa deben aceptar valores NULL. No se puede especificar para las tablas que tienen desencadenadores INSTEAD OF UPDATE.

**SET DEFAULT**

Todos los valores que forman la clave externa se establecen en los valores predeterminados si se actualiza o elimina la fila correspondiente de la tabla primaria. Para que esta restricción se ejecute, todas las columnas de clave externa deben tener definiciones predeterminadas. Si una columna acepta valores NULL y no se ha establecido un valor predeterminado explícito, NULL se convierte en el valor predeterminado explícito de dicha columna. No se puede especificar para las tablas que tienen desencadenadores INSTEAD OF UPDATE.

**CASCADE, SET NULL, SET DEFAULT** y **NO ACTION** se pueden combinar en las tablas con relaciones referenciales entre sí. Si el Motor de base de datos detecta NO ACTION, detiene y revierte las acciones CASCADE, SET NULL y SET DEFAULT relacionadas. Cuando una instrucción DELETE hace que se combinen las acciones CASCADE, SET NULL, SET DEFAULT y NO ACTION, todas las acciones CASCADE, SET NULL y SET DEFAULT se aplican antes de que el Motor de base de datos compruebe la existencia de NO ACTION.

Obtenido de: https://docs.microsoft.com/es-es/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-2017

# **B. PL/SQL**

# ¿Qué es PL/SQL?

SQL extendido

Gestor que va incorporado en Oracle para solventar las grandes limitaciones que teníamos con los editores SQL, en cuanto al control de las secuencias de ejecución de instrucciones, el uso de variables, la gestión de posibles errores, etc.

Este lenguaje está basado en ADA, por lo que incluye todas las características de los lenguajes de tercera generación. Esto nos permite manejar las variables, tener una estructura modular (procedimientos y funciones) y controlar las excepciones. Además incorpora un completo soporte para la programación orientada a objetos (POO).

Los programas creados con PL/SQL los podemos almacenar en nuestra base de datos como cualquier otro objeto quedando disponibles para los usuarios. El uso del lenguaje PL/SQL es imprescindible para construir disparadores de bases de datos (triggers).

PL/SQL está incluido en el servidor y en algunas herramientas de cliente. Soporta todos los comandos de consulta y manipulación de datos, aportando al lenguaje SQL las estructuras de control y otros elementos propios de los lenguajes de programación de 3º generación.

La unidad de trabajo en PL/SQL es el bloque, constituido por un conjunto de declaraciones, instrucciones y mecanismos de gestión de errores y excepciones.

Obtenido de: https://desarrolloweb.com/articulos/introduccion-al-lenguaje-plsql.html

# ¿Qué motores lo soportan?

SQL server, MYSQL, ORACLE Y POSTGRESQL

# **C. Datos e instrucciones en PL/SQL**

# ¿Cuáles son los tipos de datos que ofrece?

**\* NUMBER :** Almacena números enteros o de punto flotante, virtualmente de cualquier longitud, aunque puede ser especificada la precisión (Número de dígitos) y la escala que es la que determina el número de decimales.

**\* CHAR :** Almacena datos de tipo caracter con una longitud máxima de 32767 y cuyo valor de longitud por default es VARCHAR2 (Caracter de longitud variable): Almacena datos de tipo caracter empleando sólo la cantidad necesaria aún cuando la longitud máxima sea mayor.

**\* VARCHAR2:** Almacena datos de tipo caracter empleando sólo la cantidad necesaria aún cuando la longitud máxima sea mayor.

**\* BOOLEAN :** Se emplea para almacenar valores TRUE o FALSE.

**\* DATE :** Almacena datos de tipo fecha. Las fechas se almacenan internamente como datos numéricos, por lo que es posible realizar operaciones aritméticas con ellas.

**\* Atributos de tipo.** Un atributo de tipo PL/SQL es un modificador que puede ser usado para obtener información de un objeto de la base de datos. El atributo %TYPE permite conocer el tipo de una variable, constante o campo de la base de datos. El atributo %ROWTYPE permite obtener los tipos de todos los campos de una tabla de la base de datos, de una vista o de un cursor.

\* PL/SQL también permite la creación de tipos personalizados (registros) y colecciones(tablas de PL/SQL), que veremos en sus apartados correspondientes.

# ¿Cuál es la forma de definir constantes y variables?

Con “:=”

# ¿Cómo se define una variable con un tipo tomado de la base de datos?

En un bloque anónimo se utilizará la palabra DECLARE para declarar las variables y sus tipos:

DECLARE

v\_fecha DATE;

v\_numero NUMBER(2):=10;

v\_nombre VARCHAR2(50):='Laura';

constante CONSTANT NUMBER:=100;

BEGIN

dbms\_output.put\_line('Aquí vendría el código del programa');

END;

# ¿Cuál es la forma de los diferentes tipos de asignación? (Son tres)

Tipos de variables escalares: Los principales tipos de variables escalares en Oracle PL/SQL son los siguientes:

VARCHAR2(tamaño) NUMBER[(precisión,escala)] DATE CHAR(tamaño) LONG BOOLEAN BINARY\_INTEGER

Variables de tipo HOST: Las variables No-PL/SQL o de tipo HOST son variables globales que pueden usarse tanto dentro como fuera de los bloques PL/SQL.

**Atributo %TIPE:** El atributo %TYPE sirve para declarar una variable a partir de otras declaradas previamente, o a partir del tipo de una columna de la base de datos. Es decir, copiar el tipo de otra variable.

# **D. Cursores**

# ¿Qué es un cursor implícito? ¿Para qué sirve?

Los cursores implícitos se utilizan para realizar consultas SELECT que devuelven un único registro.

\* Con cada cursor implícito debe existir la palabra clave INTO.

\* Las variables que reciben los datos devueltos por el cursor tienen que contener el mismo tipo de dato que las columnas de la tabla.

\* Los cursores implícitos solo pueden devolver una única fila. En caso de que se devuelva más de una fila (o ninguna fila) se producirá una excepción. No se preocupe si aún no sabe que es una excepción, le valdrá conocer que es el medio por el que PL/SQL gestiona los errores.

# ¿Qué es un cursor explícito? ¿Para qué sirve?

Los cursores explícitos se emplean para realizar consultas SELECT que pueden devolver cero filas, o más de una fila.

\* Declarar el cursor.

\* Abrir el cursor con la instrucción OPEN.

\* Leer los datos del cursor con la instrucción FETCH.

\* Cerrar el cursor y liberar los recursos con la instrucción CLOSE.

# ¿Cuáles son las excepciones propias de uso de estos cursores?

**\* NO\_DATA\_FOUND:** Se produce cuando una sentencia SELECT intenta recuperar datos pero ninguna fila satisface sus condiciones. Es decir, cuando "no hay datos".

**\* TOO\_MANY\_ROWS:** Dado que cada cursor implícito sólo es capaz de recuperar una fila , esta excepción detecta la existencia de más de una fila.

# **E. Modularidad**

# ¿Cuál es la estructura general de un bloque PL/SQL?

Los bloques PL/SQL presentan una estructura específica compuesta de tres partes bien diferenciadas:

* **La sección declarativa** en donde se declaran todas las constantes y variables que se van a utilizar en la ejecución del bloque. Esta sección es opcional.
* **La sección de ejecución** que incluye las instrucciones a ejecutar en el bloque PL/SQL. Estas instrucciones pueden ser tanto de tipo DML como DDL, así como órdenes procedimentales. Esta es la única sección que es obligatoria.
* **La sección de excepciones** en donde se definen los manejadores de errores que soportará el bloque PL/SQL. Esta sección es opcional y no se ejecutará a menos que aparezca un error.

Cada una de las partes anteriores se delimita por una palabra reservada, de modo que un bloque PL/SQL se puede representar como sigue:

[**DECLARE**

Declaración de variables] */\*Parte declarativa\*/*

**BEGIN**

Sentencias **SQL** y PL**/SQL** */\*Parte de ejecución\*/*

[**EXCEPTION**

Manejadores de excepciones] */\*Parte de excepciones\*/*

**END**;

Un ejemplo de bloque PL/SQL sería el siguiente:

|  |  |
| --- | --- |
| [1](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n1)  [2](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n2)  [3](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n3)  [4](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n4)  [5](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n5)  [6](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n6)  [7](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n7)  [8](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n8)  [9](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n9)  [10](http://www.webtutoriales.com/articulos/estructura-de-un-bloque-pl-sql#n10) | DECLARE  v\_variable VARCHAR2(2);  BEGIN  SELECT nombre\_columna  INTO v\_variable  FROM nombre\_tabla;  EXCEPTION  WHEN exception\_name THEN  dbms\_output.put\_line('Se ha producido una excepción');  END; |

Este programa selecciona un campo de la base de datos y lo guarda en una variable. Si se produce algún error, salta la excepción

# ¿Para qué sirven las diferentes estructuras modulares? (bloque anónimo, procedimiento, función y disparador)

Los bloques PL/SQL son de varios tipos:

* **Anónimos** (Anonymous blocks). Se construyen de forma dinámica y se ejecutan una sola vez.
* **Con nombre** (Named blocks). Son bloques con nombre, que al igual que el anterior se construyen, generalmente, de forma dinámica y se ejecutan una sola vez.
* **Subprogramas**. Procedimientos, paquetes o funciones almacenados en la BD. No suelen cambiar después de su construcción y se ejecutan múltiples veces mediante una llamada call.
* **Disparadores(Triggers)**. Son bloques con nombre que también se almacenan en la BD. Tampoco suelen cambiar después de su construcción y se ejecutan varias veces. Se ejecutan de forma automática ante algún suceso de disparo, que será una orden del lenguaje de manipulación de datos *(INSERT, UPDATE o DELETE)* que se ejecuta sobre una tabla de la BD.

Un bloque anónimo es aquel bloque que no tiene asignado un nombre.

**SET** SERVEROUTPUT **ON**;

**DECLARE**

A VARCHAR(10) :**=** '';

**BEGIN**

**SELECT** TO\_CHAR(SYSDATE) **INTO** A **FROM** DUAL;

DBMS\_OUTPUT.PUT\_LINE('LA FECHA ACTUAL ES : ' **||** A);

**EXCEPTION**

**WHEN** OTHERS **THEN** DBMS\_OUTPUT.PUT\_LINE('HOLA');

**END**;

Para que la salida pueda verse al ejecutar el programa tiene que estar activa la siguiente variable:

**SET** SEVEROUTPUT **ON**;

Para mostrar el contenido de una expresión se debe utilizar la sentencia:

DBMS\_OUTPUT.PUT\_LINE (cadena\_caracteres);

Procedimientos: Se trata del programa más utilizado en PL/SQL y su formato es el siguiente:

Estructura de un bloque anónimo:

[DECLARE]

BEGIN

<instrucciones>;

[EXCEPTION]

END

Estructura de un bloque dentro de un procedimiento:

PROCEDURE nombreProcedimiento

[(<lista de parámetros>)]

IS

[DECLARE]

BEGIN

<instrucciones>;

[EXCEPTION]

<excepciones>;]

END;

Estructura de un bloque dentro de una función:

FUNCTION nombreFuncion

RETURN tipodedato

IS

[DECLARE]

BEGIN

<instrucciones>;

RETURN valor;

[EXCEPTION]

END;

Obtenido de:

<https://elbauldelprogramador.com/bloques-plsql/>