**SQL DINAMICO**

SQL Dinámico se utiliza para la ejecución de sentencias SQL o bloques PL/SQL a partir de una cadena de texto montadas de manera dinámica.

Esto nos permite poder cambiar el contenido de la ejecución de objetos sin necesidad de volver a realizar su compilación.

También nos permite utilizar SGA y reducir los pasos del servidor parse mediante variables BIND.

Tenemos dos formas de crear SQL dinámico:

1. Mediante instrucciones de texto UPDATE, DELETE o INSERT con variables BIND.
2. Mediante el comando execute inmediate, que nos permite todo lo anterior y, además, incluir cursores en la consulta.

El uso de execute immediate es muy cómodo para la construcción de sql dinámicos, el problema es el posible uso de SQL Injection en PL/SQL con dichas sentencias.

Las bind variables, además de evitar parseos innecesarios, evitan SQL Injection en PL/SQL dinámico, por lo que es importante utilizar bind variables en vez de concatenación de cadenas.

VENTAJAS:

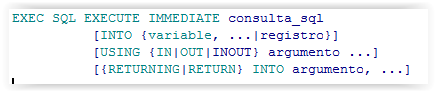
* Creación de sentencias en tiempo de ejecución
* Interactivo con el usuario
* Acceder a objetos no existentes en tiempo de compilación
* Gestión de permisos de usuarios de forma dinámica
* Permite ejecutar instrucciones DDL (create, alter, drop, grant, …)

DESVENTAJAS:

* No siempre se forman las consultas más óptimas
* Problemas de seguridad por ataques de inyección SQL

SINTAXIS:

**EXECUTE IMMEDIATE** es el sustituto del paquete **DBMS\_SQL**



consulta\_sql: instrucción SQL o bloque PL/SQL

variable: variable donde se almacena el valor de la columna seleccionada

registro: variable estructurada, en la que se obtiene una fila

USING: Sirve para pasar parámetros a las sentencias. bind\_arg: son variables o expresiones que contienen los parámetros de la sentencia.

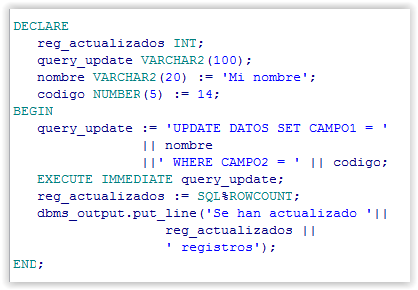
Cuando se ejecuta la sentencia, se reemplaza cada variable en la sentencia sql (un identificador con dos puntos delante, como :valor1) por su correspondiente valor según su posición.

Se pueden pasar números, fechas y cadenas.

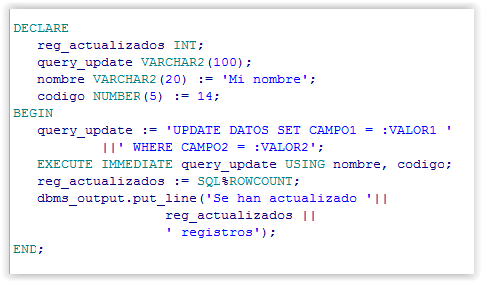
No se pueden pasar booleanos o literales de NULL pero si se puede pasar una variable a nulo con el tipo de dato adecuado.

EJEMPLOS:

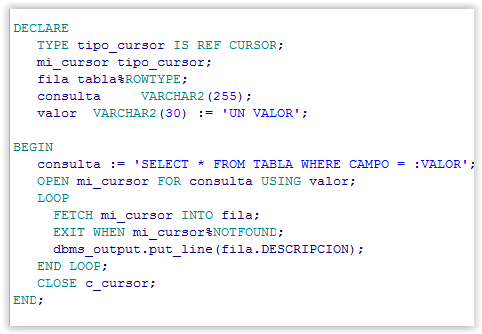
Ejemplo de UPDATE sin bind variables.



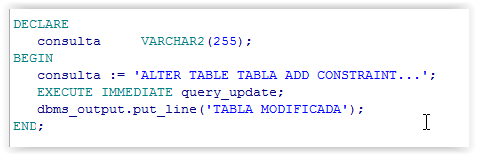
Ejemplo de UPDATE utilizando bind variables, mucho más eficiente:



Ejemplo de CURSOR utilizando bind variables:

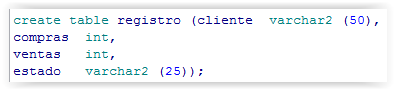


Ejemplo para consultas DDL:

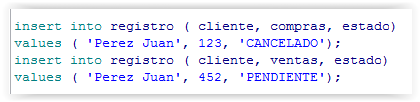


Vamos a visualizar un ejemplo de la utilidad de las consultas dinámicas.

Si poseemos la siguiente estructura de una tabla:



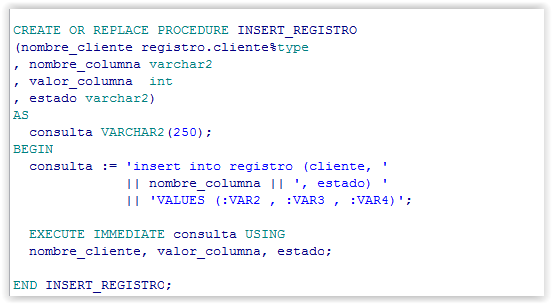
En la que deseamos incluir los siguientes registros, en donde se guardará información de ventas o compras de un cliente, tomando en cuenta que por cada transacción, solo puede existir un valor en compras o ventas, pero no los dos, los datos son ingresados de la siguiente manera:

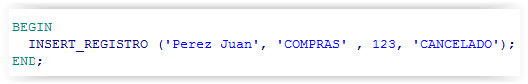


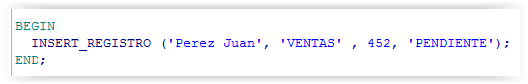
Como podemos ver la única diferencia entre estas sentencias de INSERT es la columna a la cual hacen referencia.

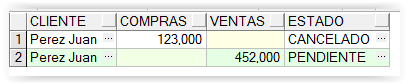
Tendríamos que hacer dos procedimientos diferentes o bien una sentencia de control.

Con SQL Dinámico, tenemos la posibilidad de realizar la acción completa sin sentencias de control.









Realizar un procedimiento para crear una Primary Key en cualquier tabla:

CREATE OR REPLACE PROCEDURE CREARPK

(TABLA VARCHAR2

, PK\_NOMBRE VARCHAR2

, CAMPO VARCHAR2)

AS

consulta VARCHAR2(200);

BEGIN

*--ALTER TABLE EMP*

*--ADD CONSTRAINT PK\_EMP*

*--PRIMARY KEY (EMP\_NO)*

consulta := 'ALTER TABLE ' || TABLA

|| ' ADD CONSTRAINT ' || PK\_NOMBRE

|| ' PRIMARY KEY (' || CAMPO || ')';

EXECUTE IMMEDIATE consulta;

DBMS\_OUTPUT.PUT\_LINE(consulta);

END;

BEGIN

CREARPK('EMP', 'PK\_EMP', 'EMP\_NO');

END;

------------------------------------------------------

BEGIN

CREARPK('DEPT', 'PK\_DEPT', 'DEPT\_NO');

END;