



ORACLE PL/SQL



MÓDULO 08

SQL EN PL/SQL

GUSTAVO CORONEL

desarrollasoftware.com



ORACLE PL/SQL



Contenido

CATEGORÍAS DEL LENGUAJE SQL	3
Uso de SQL en PL/SQL.....	3
SQL DINÁMICO	4
SENTENCIA: SELECT	5
SENTENCIA: INSERT.....	6
SENTENCIA: UPDATE.....	8
SENTENCIA: DELETE	10
NOMBRES DE VARIABLES.....	11
CLÁUSULA RETURNING	12
REFERENCIAS DE TABLAS	13
ENLACES DE BASE DE DATOS	14
SINÓNIMOS	15
CURSOS VIRTUALES	16
JAVA ORIENTADO A OBJETOS.....	¡ERROR! MARCADOR NO DEFINIDO.
PROGRAMACIÓN DE BASE DE DATOS CON JAVA JDBC	¡ERROR! MARCADOR NO DEFINIDO.



ORACLE PL/SQL



CATEGORÍAS DEL LENGUAJE SQL

1. DML Lenguaje de Manipulación de Datos: Select, Insert, Update, Delete.
2. DDL Lenguaje de Definición de Datos: Create, Alter, Drop, Grant.
3. Control de Transacciones: Commit, Rollback.
4. Control de Sesiones: Alter Session.
5. Control del Sistema: Alter System.

Uso de SQL en PL/SQL

Las categorías permitidas de SQL en PL/SQL son solamente la 1 y 3: DML y Control de Transacciones.



ORACLE PL/SQL



SQL DINÁMICO

Permite ejecutar cualquier tipo de instrucción SQL desde PL/SQL.

Script 1

```
create or replace procedure pr107( cmd varchar2)
is
begin
    execute immediate cmd;
end;
```

Ejecución 1:

```
SQL> exec pr107('create table t1 ( id number, dato varchar2(30) )');

PL/SQL procedure successfully completed.
```

Ejecución 2:

```
SQL> exec pr107('insert into t1 values( 1, ''Oracle is Powerful'' )');

PL/SQL procedure successfully completed.
```

Verificando la tabla T1:

```
SQL> select * from t1;

      ID DATO
-----
      1 Oracle is Powerful
```



ORACLE PL/SQL



SENTENCIA: SELECT

Sintaxis

```
Select columnas into variables/registro  
From NombreTabla  
where condición;
```

Script 2

Consultar la cantidad de empleados y el importe de la planilla de un departamento.

```
create or replace procedure pr108(cod dept.deptno%type)  
is  
    emps number;  
    planilla number;  
begin  
    select count(*), sum(sal) into emps, planilla  
    from emp  
    where deptno = cod;  
    dbms_output.put_line('Empleados: ' || emps);  
    dbms_output.put_line('Planilla: ' || planilla);  
end;
```

Ejecución:

```
SQL> exec pr108( 20 );  
Empleados: 5  
Planilla: 10875  
  
PL/SQL procedure successfully completed.
```



ORACLE PL/SQL



SENTENCIA: INSERT

Sintaxis 1

```
Insert into NombreTabla[(columnas)] values(datos);
```

Sintaxis 2

```
Insert into NombreTabla[(columns)] select ... ;
```

Script 3

Procedimiento para registrar un nuevo departamento.

```
create or replace procedure pr109( cod number, nom varchar2, loc varchar2)
is
begin
    insert into dept values(cod, nom, loc);
    commit;
    dbms_output.put_line('Proceso OK');
end;
```

Ejecución:

```
SQL> exec pr109( 50, 'Deportes', 'Los Olivos' );
Proceso OK

PL/SQL procedure successfully completed.
```



ORACLE PL/SQL



Verificando tabla **DEPT**:

```
SQL> select * from dept;
```

DEPTNO	DNAME	LOC

10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	Deportes	Los Olivos



SENTENCIA: UPDATE

Sintaxis

```
Update Nombretabla  
Set  columna1 = valor,  
     (columna2, columna3, ...) = (sentencia_select),  
     ...  
where (condición);
```

Script 4

Para este ejemplo crearemos la tabla resumen, donde almacenaremos el número de empleados por departamento y el importe de su planilla.

Creación de la tabla resumen:

```
SQL> create table resumen(  
2   deptno number(2) primary key,  
3   emps number(2),  
4   planilla number(10,2)  
5 );
```

Table created.

Insertando los códigos de los departamentos:

```
SQL> insert into resumen(deptno)  
2   select deptno from dept;
```

5 rows created.



ORACLE PL/SQL



Procedimiento para actualizar las columnas **EMPS** y **PLANILLA**:

```
create or replace procedure pr110
is
begin
    update resumen
        set (emps,planilla) = (select count(*), sum(sal) from emp
                                where emp.deptno = resumen.deptno);
    commit;
    dbms_output.put_line('Proceso Ok');
end;
```

Ejecución:

```
SQL> exec pr110;
Proceso Ok

PL/SQL procedure successfully completed.
```

Verificar la ejecución:

```
SQL> select * from resumen;
```

DEPTNO	EMPS	PLANILLA
10	3	8750
20	5	10875
30	6	9400
40	0	
50	0	



ORACLE PL/SQL



SENTENCIA: DELETE

Sintaxis:

```
Delete from NombreTabla  
where condición;
```

Script 5

Desarrollar un procedimiento para eliminar un departamento, primero debe verificar que no tenga registros relacionados en la tabla emp.

```
create or replace procedure pr11(cod number)  
is  
    cont number;  
begin  
    select count(*) into cont from dept where deptno = cod;  
    if cont = 0 then  
        dbms_output.put_line('No existe');  
        return;  
    end if;  
    select count(*) into cont from emp where deptno = cod;  
    if cont > 0 then  
        dbms_output.put_line('No puede se eliminado');  
        return;  
    end if;  
    delete from dept where deptno = cod;  
    commit;  
    dbms_output.put_line('Proceso Ok');  
end;
```

Ejecución:

```
SQL> exec pr11(10);  
No puede se eliminado  
  
PL/SQL procedure successfully completed.
```



NOMBRES DE VARIABLES

Se recomienda no utilizar nombres de variables y/o parámetros iguales a los nombres de las columnas, sobre todo si van a ser utilizadas en las instrucciones SQL.

Script 6

Desarrollar un procedimiento para eliminar un empleado. El siguiente procedimiento tiene un resultado inesperado.

```
create or replace procedure pr112(empno number)
is
begin
    delete from emp where empno = empno;
end;
```

Si intentamos eliminar un empleado, se eliminarán todos los registros de la tabla **EMP**:

```
SQL> exec pr112(7654);

PL/SQL procedure successfully completed.
```

Verifiquemos el resultado:

```
SQL> select * from emp;

no rows selected
```

Esto sucede incluso si el código del empleado no existe. Ahora ejecutemos la sentencia **rollback** para recuperar los empleados.

```
SQL> rollback;

Rollback complete.
```

Si consultamos nuevamente la tabla **EMP** tendremos los registros recuperados.



CLÁUSULA RETURNING

Sirve para obtener información de la última fila modificada, puede ser utilizado con las sentencias INSERT, UPDATE o DELETE.

Sintaxis:

```
returning expresión, ... into variable, ... ;
```

Script 7

Ilustración de Returning. La tabla TEST y la secuencia SQTEST se crean en la lección 3.

```
create or replace procedure pr113(msg varchar2)
is
  v_rowid rowid;
  v_id    number;
begin
  insert into test values(sqtest.nextval,msg)
    returning rowid, id into v_rowid, v_id;
  commit;
  dbms_output.put_line('RowId: ' || v_rowid);
  dbms_output.put_line('Id:    ' || v_id);
end;
```

Ejecución:

```
SQL> exec pr113('El deporte es salud.');
```

RowId:	AAQAEJAABAAA
Id:	21



REFERENCIAS DE TABLAS

Sintaxis:

```
[esquema.]tabla[@enlace]
```

Script 8

En el siguiente script iniciamos sesión con privilegio **sysdba** y luego consultamos la tabla **dept** de **scott**.

```
SQL> conn / as sysdba  
Conectado.
```

```
SQL> select * from scott.dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON



ORACLE PL/SQL



ENLACES DE BASE DE DATOS

Sintaxis

```
create [shared] [public] database link nombre_enlace
  connect to nombre_usuario identified by contraseña
  [using cadena_sqlnet];
```

Script 9

Crearemos un enlace remoto público para conectarnos como usuario **HR**, este enlace debe ser creado como usuario **SYS**:

```
SQL> conn / as sysdba
Connected.

SQL> create public database link lnk_demo
  2   connect to hr identified by hr
  3   using 'dbegcc';

Database link created.
```

Ahora haremos una consulta al esquema **HR**:

```
SQL> conn scott/tiger
Connected.

SQL> select employee_id, first_name
  2   from employees@lnk_demo
  3   where department_id = 30;

EMPLOYEE_ID FIRST_NAME
-----
114 Den
115 Alexander
116 Shelli
117 Sigal
118 Guy
119 Karen

6 rows selected.
```



ORACLE PL/SQL



SINÓNIMOS

Facilitan la referencia a tablas de otros esquemas, incluso de otros servidores.

Sintaxis

```
create [or replace] [public] synonym [esquema.] sinonimo
for [esquema.] objeto [@dblink]
```

Script 10

Crearemos un sinónimo público para acceder a la tabla **EMPLOYEEES** del esquema **HR**. Debemos crearlo con **SYS**.

```
SQL> conn / as sysdba
Connected.
SQL> create or replace public synonym hr_emp
2     for employees@lnk_demo;

Synonym created.
```

Ahora como **scott** accederemos a la tabla **employees** mediante el sinónimo:

```
SQL> conn scott/tiger
Connected.

SQL> select employee_id, first_name
2     from hr_emp
3     where rownum = 1;

EMPLOYEE_ID FIRST_NAME
-----
100 Steven
```



ORACLE PL/SQL



CURSOS VIRTUALES

En estos enlaces se publican cupones de descuento:

- <https://github.com/gcoronelc/UDEMY>
- <https://www.facebook.com/groups/bolsa.sistemas>
- <https://www.facebook.com/groups/universidadjava>
- <https://www.facebook.com/groups/desarrollasoftware>
- <https://chat.whatsapp.com/H6BBebNDZHEAqAk6gTU5ZS>

JAVA ORIENTADO A OBJETOS



JAVA ORIENTADO A OBJETOS

Eric Gustavo Coronel Castillo
www.desarrollasoftware.com
INSTRUCTOR

<https://bit.ly/2B3ixUW>

PROGRAMACIÓN DE BASE DE DATOS CON JAVA JDBC



PROGRAMACIÓN DE BASE DE DATOS ORACLE CON JAVA JDBC

Eric Gustavo Coronel Castillo
www.desarrollasoftware.com
INSTRUCTOR

<https://bit.ly/31apy0O>