Este post es un resumen de PL/SQL. Pensado para las personas que estén preparándose los exámenes de grado superior a modo de guía rápida.

PL/SQL nos permite un pequeño ámbito de programación dentro de las bases de datos. Enfocado a administración, eventos o tareas en la misma.

La éxtasis para ejecutar un código arbitrario:

```
DECLARE
   <declarations section>
BEGIN
   <executable command(s)>
[EXCEPTION
   exception_section]
END;
```

Un hola mundo seria:

```
DECLARE
   message  varchar2(20):= 'Hello, World!';
BEGIN
   dbms_output.put_line(message);
END;
/
```

Se usa “/” para declarar que hemos terminado la ejecución de código.

Un delimitador es un símbolo con un significado especial. La siguiente es la lista de delimitadores en PL/SQL:

Delimiter	Description
+, -, *, /	Addition, subtraction/negation, multiplication, division
%	Attribute indicator
'	Character string delimiter
.	Component selector
(,)	Expression or list delimiter
:	Host variable indicator
,	Item separator
"	Quoted identifier delimiter
=	Relational operator
@	Remote access indicator
;	Statement terminator
:=	Assignment operator
=>	Association operator
||	Concatenation operator
**	Exponentiation operator
<<, >>	Label delimiter (begin and end)
/*, */	Multi-line comment delimiter (begin and end)
--	Single-line comment indicator
..	Range operator
<, >, <=, >=	Relational operators
<>, '=, ~=, ^=	Different versions of NOT EQUAL

Como en cualquier lenguaje tendremos procedimientos:

```
---- Procedimiento
-- Sintaxis
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
  < procedure_body >
[EXCEPTION
   exception_section]
END procedure_name;
-- Ejemplo
CREATE OR REPLACE PROCEDURE greetings
AS
BEGIN
   dbms_output.put_line('Hello World!');
END;
/
```

Y funciones:

```
---- Función
-- Sintaxis
CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
   < function_body >
```

```
END [function_name];
-- Ejemplo
CREATE OR REPLACE FUNCTION totalCustomers
RETURN number IS
   total number(2) := 0;
BEGIN
   SELECT count(*) into total
   FROM customers;
   RETURN total;
END;
```

Lo mejor de PL/SQL es que permite controlar los datos a través de los eventos o triggers:

```
---- Triggers
-- Sintaxis
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
   Declaration-statements
BEGIN
   Executable-statements
EXCEPTION
   Exception-handling-statements
END;
-- Ejemplo
```

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
   sal_diff number;
BEGIN
   sal_diff := :NEW.salary  - :OLD.salary;
   dbms_output.put_line('Old salary: ' || :OLD.salary);
   dbms_output.put_line('New salary: ' || :NEW.salary);
   dbms_output.put_line('Salary difference: ' || sal_diff);
END;
```

Se recomienda la lectura del siguiente link, con toda la documentación oficial en ingles.