Sintaxis de `CREATE DATABASE`

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification [, create_specification] ...]

create_specification:
    [DEFAULT] CHARACTER SET charset_name
  | [DEFAULT] COLLATE collation_name
```

Sintaxis de `CREATE TABLE`

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)]
    [table_options] [select_statement]
```

O:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [() LIKE old_tbl_name ()];

create_definition:
    column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type]
(index_col_name,...)
  | KEY [index_name] [index_type] (index_col_name,...)
  | INDEX [index_name] [index_type] (index_col_name,...)
  | [CONSTRAINT [symbol]] UNIQUE [INDEX]
        [index_name] [index_type] (index_col_name,...)
  | [FULLTEXT|SPATIAL] [INDEX] [index_name] (index_col_name,...)
  | [CONSTRAINT [symbol]] FOREIGN KEY
        [index_name] (index_col_name,...) [reference_definition]
  | CHECK (expr)

column_definition:
    col_name type [NOT NULL | NULL] [DEFAULT default_value]
        [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
        [COMMENT 'string'] [reference_definition]

type:
    TINYINT[(length)] [UNSIGNED] [ZEROFILL]
  | SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
  | MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
  | INT[(length)] [UNSIGNED] [ZEROFILL]
  | INTEGER[(length)] [UNSIGNED] [ZEROFILL]
  | BIGINT[(length)] [UNSIGNED] [ZEROFILL]
  | REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
```

```
  | DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
  | NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
  | DATE
  | TIME
  | TIMESTAMP
  | DATETIME
  | CHAR(length) [BINARY | ASCII | UNICODE]
  | VARCHAR(length) [BINARY]
  | TINYBLOB
  | BLOB
  | MEDIUMBLOB
  | LONGBLOB
  | TINYTEXT [BINARY]
  | TEXT [BINARY]
  | MEDIUMTEXT [BINARY]
  | LONGTEXT [BINARY]
  | ENUM(value1,value2,value3,...)
  | SET(value1,value2,value3,...)
  | spatial_type

index_col_name:
    col_name [(length)] [ASC | DESC]

reference_definition:
    REFERENCES tbl_name [(index_col_name,...)]
               [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
               [ON DELETE reference_option]
               [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION

table_options: table_option [table_option] ...

table_option:
    {ENGINE|TYPE} = engine_name
  | AUTO_INCREMENT = value
  | AVG_ROW_LENGTH = value
  | [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]
  | CHECKSUM = {0 | 1}
  | COMMENT = 'string'
  | MAX_ROWS = value
  | MIN_ROWS = value
  | PACK_KEYS = {0 | 1 | DEFAULT}
  | PASSWORD = 'string'
  | DELAY_KEY_WRITE = {0 | 1}
  | ROW_FORMAT =
{DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}
  | RAID_TYPE = { 1 | STRIPED | RAID0 }
        RAID_CHUNKS = value
        RAID_CHUNKSIZE = value
  | UNION = (tbl_name[,tbl_name]...)
  | INSERT_METHOD = { NO | FIRST | LAST }
  | DATA DIRECTORY = 'absolute path to directory'
  | INDEX DIRECTORY = 'absolute path to directory'
```

```
select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select
statement)
```

Sintaxis de `CREATE INDEX`

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
    [USING index_type]
    ON tbl_name (index_col_name,...)

index_col_name:
    col_name [(length)] [ASC | DESC]
```

Sintaxis de `ALTER TABLE`

```
ALTER [IGNORE] TABLE tbl_name
    alter_specification [, alter_specification] ...

alter_specification:
    ADD [COLUMN] column_definition [FIRST | AFTER col_name ]
  | ADD [COLUMN] (column_definition,...)
  | ADD INDEX [index_name] [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        PRIMARY KEY [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [index_name] [index_type] (index_col_name,...)
  | ADD [FULLTEXT|SPATIAL] [index_name] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [index_name] (index_col_name,...)
        [reference_definition]
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  | CHANGE [COLUMN] old_col_name column_definition
        [FIRST|AFTER col_name]
  | MODIFY [COLUMN] column_definition [FIRST | AFTER col_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP INDEX index_name
  | DROP FOREIGN KEY fk_symbol
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col_name
  | CONVERT TO CHARACTER SET charset_name [COLLATE
collation_name]
  | [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | table_options
```

Sintaxis de `DROP TABLE`

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

Sintaxis de `DROP DATABASE`

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

Sintaxis de `RENAME TABLE`

```
RENAME TABLE tbl_name TO new_tbl_name
    [, tbl_name2 TO new_tbl_name2] ...
```

Sintaxis de `DELETE`

Sintaxis para una tabla:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
[WHERE where_definition]
[ORDER BY ...]
[LIMIT row_count]
```

Sintaxis para múltiples tablas:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
tbl_name[.*] [, tbl_name[.*] ...]
FROM table_references
[WHERE where_definition]
```

O:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
FROM tbl_name[.*] [, tbl_name[.*] ...]
USING table_references
[WHERE where_definition]
```

Sintaxis de `UPDATE`

Sintaxis para una tabla:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
SET col_name1=expr1 [, col_name2=expr2 ...]
[WHERE where_definition]
[ORDER BY ...]
[LIMIT row_count]
```

Sintaxis para múltiples tablas:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
SET col_name1=expr1 [, col_name2=expr2 ...]
[WHERE where_definition]
```

Sintaxis de `SELECT`

```
SELECT
[ALL | DISTINCT | DISTINCTROW ]
[HIGH_PRIORITY]
[STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr, ...
[INTO OUTFILE 'file_name' export_options
| INTO DUMPFILE 'file_name']
[FROM table_references
[WHERE where_definition]
[GROUP BY {col_name | expr | position}
[ASC | DESC], ... [WITH ROLLUP]]
[HAVING where_definition]
[ORDER BY {col_name | expr | position}
[ASC | DESC] , ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[PROCEDURE procedure_name(argument_list)]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

JOIN

```
table_reference, table_reference
table_reference [INNER | CROSS] JOIN table_reference
[join_condition]
table_reference STRAIGHT_JOIN table_reference
table_reference LEFT [OUTER] JOIN table_reference join_condition
table_reference NATURAL [LEFT [OUTER]] JOIN table_reference
{ ON table_reference LEFT OUTER JOIN table_reference
ON conditional_expr }
table_reference RIGHT [OUTER] JOIN table_reference join_condition
table_reference NATURAL [RIGHT [OUTER]] JOIN table_reference
```

table_reference se define como:

```
  tbl_name [[AS] alias]
  [[USE INDEX (key_list)]
  | [IGNORE INDEX (key_list)]
  | [FORCE INDEX (key_list)]]
```

Sintaxis de UNION

```
SELECT ...
UNION [ALL | DISTINCT]
SELECT ...
[UNION [ALL | DISTINCT]
SELECT ...]
```

Sintaxis de una Vista

```
CREATE VIEW Nombrevista [(columna [,columna])] AS consulta
[WITH {CHECK OPTION | READ ONLY}
Nombrevista es el nombre que damos a la vista.
[(columna [,columna])] son los nombres de columnas que va a
contener la vista. Si no se
ponen, se asumen los nombres de columna devueltos por la
consulta.
AS consulta determina las columnas y las tablas que aparecerán
en la vista.
[WITH CHECK OPTION] es la opción de comprobación para una vista.
Si se especifica, SQL
comprueba automáticamente cada operación INSERT y UPDATE sobre
la vista para
asegurarse que las filas resultantes satisfagan el criterio de
búsqueda de la definición de
la vista. Si la fila insertada o modificada no satisface la
condición de creación de la vista,
la sentencia INSERT o UPDATE falla y no se realiza la operación.
[WITH READ ONLY] especifica que sólo se puede hacer SELECT de
las filas de la vista.
[CONSTRAINT nombrerestriccion]
```

Sintaxis de PL / SQL

```
DECLARE
       CURSOR cur_emp IS SELECT num_empleado, nomb_empleado FROM
empleado;
BEGIN
FOR emp_registro IN cur_emp LOOP -- Apertura y declaración del
cursor ímplitamente
IF emp_registro.edad>30
```

```
        THEN ...
        END LOOP; -- Cierre implícito del cursor
END;
```

Excepciones Oracle PL / SQL

Existe un conjunto de excepciones predefinidas por Oracle cuyos nombres aparecen a continuación agrupados por su funcionalidad:

- **NO_DATA_FOUND, TOO_MANY_ROWS**. Ocurren cuando un select no selecciona nada o selecciona varias filas cuando sólo se esperaba una.
- **INVALID_NUMBER, VALUE_ERROR, ZERO_DIVIDE, DUP_VAL_ON_INDEX**. Las tres primeras situaciones se producen por operaciones invalidas de tratamiento de números, y la ultima cuando se intenta insertar una clave primaria duplicada.
- **CURSOR_ALREADY_OPEN, INVALID_CURSOR.** La primera situación ocurre al intentar abrir un cursor ya abierto, y la segunda al intentar hacer una operación invalida sobre un cursor.
- **PROGRAM_ERROR, STORAGE_ERROR, TIMEOUT_ON_RESOURCE.** Detectan errores de almacenamiento o de ejecución.

ACCESS_INTO_NULL	Se intenta asignar valores a un atributo de un objeto que contiene nulos.
CASE_NOT_FOUND	Ninguna de las opciones WHEN de una sentencia CASE ha sido seleccionada y no existe la cláusula ELSE.
COLLECTION_IS_NULL	Se intenta aplicar métodos de colección diferentes a EXISTS a una tabla anidada o un Varray y ésta contiene valores nulos o no está inicializada.
CURSOR_ALREADY_OPEN	Se intenta abrir un cursor que ya está abierto.
DUP_VAL_ON_INDEX	Se intenta guardar un valor duplicado en un índice que no permite valores duplicados.
INVALID_CURSOR	Se intenta realizar una operación sobre un cursor que está cerrado.
INVALID_NUMBER	En un comando SQL la conversión de una cadena alfanumérica a un número es incorrecta ya que no representa un número válido. En PL/SQL levanta la excepción VALUE_ERROR.
LOGIN_DENIED	Se intenta conectarse a Oracle con un usuario y una contraseña incorrectos.

NO_DATA_FOUND	Un SELECT INTO no devuelve filas, o se referencia a un elemento borrado en una tabla anidada o un elemento no inicializado en una tabla indexada. Las funciones agregadas de grupo (AVG, SUM, COUNT,etc.) siempre devuelven un nulo o un cero por lo que un comando SELECT con funciones agregadas nunca levantará esta excepción. Un comando FETCH puede que no devuelva filas por lo que no levantará esta excepción en el caso de que no devuelva ninguna fila.
NOT_LOGGED_ON	Se intenta realizar una llamada a una base de datos sin estar conectado a ella.
PROGRAM_ERROR	Se ha producido un error interno de PL/SQL.
ROWTYPE_MISMATCH	A host variable (Ejemplo: variable de un programa 3GL) y la variable de un cursor PL/SQL no son del mismo tipo.
SELF_IS_NULL	Se intenta usar el método MEMBER a una instancia nula.
STORAGE_ERROR	Falta de recursos de memoria o está corrupta.
SUBSCRIPT_BEYOND_COUNT	Se intenta referenciar a un elemento de una tabla anidada o un varray utilizando un valor mayor que el número de elementos de la tabla.
SUBSCRIPT_OUTSIDE_LIMIT	Se intenta referenciar a un elemento de una tabla anidada o un varray utilizando un valor que está fuera del rango permitido (Ejemplo el valor -1).
SYS_INVALID_ROWID	La conversión de una cadena alfanumérica a un tipo rowid universal es incorrecta porque no representa un valor válido.
TIMEOUT_ON_RESOURCE	Se ha producido un time-out (exceso de tiempo) esperando un recurso.
VALUE_ERROR	Se ha producido un error en una operación aritmética, conversión, truncamiento o límite de precisión. Ejemplo: La columna que recibe un valor de un comando SELECT INTO es menor que el tamaño de la columna de la base de datos.
ZERO_DIVIDE	Se ha producido una división por cero. (No existe el valor infinito en Oracle).

Procedimientos SQL

```
CREATE [OR REPLACE]
PROCEDURE Nombre_procedimiento [(declaración de parámetros)]
      [AUTHID {DEFINER | CURRENT_USER}]
{IS | AS}
      [PRAGMA AUTONOMOUS_TRANSACTION;]
      [Declaraciones locales de tipos, variables, etc]
BEGIN
      Sentencias ejecutables del procedimiento
[EXCEPTION
      Excepciones definidas y las acciones de estas excepciones]
END [Nombre_procedimiento];
/
```

Funciones SQL

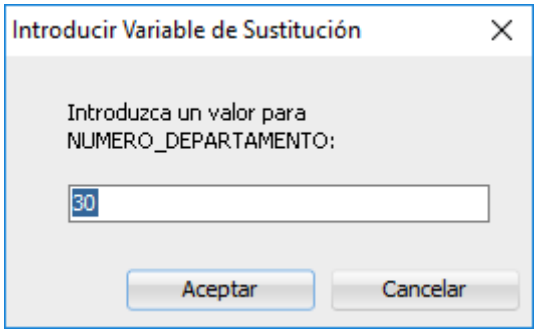
La sintaxis para crear una función es:

```
CREATE [OR REPLACE]
FUNCTION Nombre_Función
            [(declaracion_parámetro
            [, declaracion_parámetro]...)]
RETURN Tipo_dato
      [AUTHID {DEFINER | CURRENT_USER}]
{IS | AS}
      [DETERMINISTIC]
      [PRAGMA AUTONOMOUS_TRANSACTION;]
      [Declaraciones locales de tipos, variables, etc]
BEGIN
      /*Sentencias ejecutables*/
    ............
return literal/variable;
[EXCEPTION
      --Excepciones definidas y las acciones de estas excepciones]
   .........
   return literal/variable;
END [Nombre_Función];
```

Petición de valor por consola PL/SQL

Sintaxis para pedir un valor por variable:

V1 := &NUMERO_DEPARTAMENTO;



Sintaxis de un trigger os disparador

```
CREATE [OR REPLACE] TRIGGER nombretrigger
   {BEFORE | AFTER}
   {DELETE | INSERT | UPDATE [OF <lista_columnas>]}
   ON nombretabla
              [FOR EACH {STATEMENT | ROW    [WHEN (condicion)]}]
              < CUERPO DEL TRIGGER (BLOQUE PL/SQL)>
```

```
CREATE TRIGGER ...
BEFORE INSERT OR UPDATE OR DELETE ON empleados ...
BEGIN
   IF INSERTING THEN
   ...
   ELSIF DELETING THEN
   ...
   ELSIF UPDATING('salary') THEN
   ...
  END IF
  ...
END;
```

INSERTING
Devuelve TRUE si el evento que disparó el trigger fue un comando INSERT.

DELETING
Devuelve TRUE si el evento que disparó el trigger fue un comando DELETE.

UPDATING
Devuelve TRUE si el evento que disparó el trigger fue una instrucción UPDATE.

UPDATING ('nombrecolumna')
Devuelve TRUE si el evento que disparó el trigger fue una instrucción UPDATE y la columna especificada ha sido actualizada.