# SQL

(Structured Query Language)

## Índice



Nota: Para todos los ejemplos de este documento se supone que el usuario crea los objetos en su propio esquema.

# 1.- INTRODUCCIÓN

El SQL es un lenguaje orientado al manejo de bases de datos relacionales RDBMS (relational database management system)

Fue desarrollado sobre un prototipo de gestor de base de datos relacionales denominado SYSTEMR por IBM a mediados de los años setenta. En 1.979 Oracle Corporation presentó la primera implementación comercial de SQL y el instituto ANSI (American National Standard Institute) adoptó el lenguaje SQL como estándar para la gestión de base de datos relacionales en 1.986.

En 1.987 lo adopto ISO (International Standarization Organization)

El lenguaje SQL tiene varias versiones implementados por diversas alternativas de motores de bases de datos. Estas versiones tienen algunas variaciones tales como el tratamiento de mayúsculas y minúsculas, los índices, etc. Los motivos de la falta de compromiso con el el estándar SQL son variados, desde que el propio estándar no cubre todas las posibilidades del lenguaje en todos los escenarios hasta que las tecnologías de bases de datos son tan variadas que es complejo adaptarse a unas reglas únicas.

ISO ha ido publicando sucesivas revisiones lo largo del tiempo, cada dos o tres años. Algunas de las últimas en 2016 y 2019. La referente al Database Lenguaje se puede encontrar en la siguiente dirección: "https://www.iso.org/standard/76584.html".

# 2.- ÓRDENES BÁSICAS

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

SENTECIAS DDL

Son las que se encarga de la modificación de la estructura de los objetos de la base de datos. Los comandos usados son:

- CREATE: Crear objetos de base de datos
- DROP: Eliminar objetos
- ALTER: Modificar objetos
- GRANT: conceder privilegios
- REVOKE: retirar privilegios

SENTENCIAS DML

Son las que permite llevar a cabo las tareas de consulta o manipulación de los datos. Los comandos usados son:

- INSERT: Insertar filas en una tabla
- UPDATE: actualizar filas de una tabla
- DELETE: Eliminar filas de datos en una tabla
- SELECT: Recuperar filas de datos de una tabla o vista

## 3.- TIPOS DE DATOS

CHAR (n)

- Cadena de caracteres de longitud fija, tiene un tamaño n bytes.
- El tamaño máximo en BD es 2000 bytes
- El tamaño mínimo y por defecto es de 1 byte.

NCHAR(n)

Almacena un valor alfanumérico de longitud fija con posibilidad de cambio de juego de caracteres. Puede almacenar tantos caracteres ASCII, EBCDIC, UNICODE…

VARCHAR2 (n)

- Cadena de caracteres de longitud variable, tiene un tamaño máximo de n bytes.
- Es obligatorio especificar el tamaño.
- El tamaño máximo en BD es 4000 bytes
- el tamaño mínimo es de 1 byte

NVARCHAR2(n)

Almacena un valor alfanumérico de longitud variable con posibilidad de cambio de juego de caracteres. Puede almacenar tanto caracteres ASCII, EBCDIC, UNICODE…

NUMBER (p,s)

- Datos numéricos, enteros o decimales, con o sin signos.
- Número de p digitos de los cuales s son decimales.
- El tamaño de p va de 1 a 38 y el s desde -84 a 127.
- No es obligatorio especificar el tamaño.

DATE

Almacena un valor fecha y hora. En un tipo de dato DATE, Oracle almacena internamente los siguientes datos: Siglo, Año, Mes, Día, Hora, Minuto, Segundo

El formato por defecto de las fechas es dependiente del idioma instalado. Este formato puede ser alterado en cualquier momento mediante el parámetro de inicialización NLS_DATE_FORMAT.

ROWID.

Valor hexadecimal que representa la dirección única de una fila en su tabla.

El ROWID de una fila es un identificador único para una fila dentro de una base de datos. No hay dos filas con el mismo ROWID. Este tipo de dato sirve para guardar punteros a filas concretas. El ROWID se compone de:

- Número de datafile donde se almacena la fila (se pueden ver en DBA_DATA_FILES)
- Dirección del bloque donde está la fila
- Posición dentro del bloque

LONG.

Almacena cadenas de caracteres de longitud variable de hasta 2 GB. Es Proporcionado para la compatibilidad con versiones anteriores.

Posee muchas restricciones a la hora de utilizarlo. Sólo se permite una columna por tabla

CLOB.

Representa a objeto de caracteres de hasta 4 GB (sustituye a LONG)

BLOB.

Representa a objetos binarios de hasta 4 GB

RAW. (en desuso, se sustituye por los tipos lob)

Almacena datos binarios de longitud variables de hasta 2000 byte

LONG RAW. (en desuso, se sustituye por los tipos lob)

Almacena datos binarios de longitud variables de hasta 2 GB

## 4.- CREACION DE TABLAS

**CREATE TABLE** [esquema.]nombretabla
(
Columna1 Tipo_dato
[CONSTRAINT nombre_restricción] [restricciones_de_columna],
Columna2 Tipo_dato
[CONSTRAINT nombre_restricción] [restricciones_de_columna],
.....
[CONSTRAINT nombre_restricción][restricción de tabla],…
[CONSTRAINT nombre_restricción][restricción de tabla]
)

Los nombres de la tabla tienen que cumplir los siguientes requisitos:

- Deben comenzar con una letra.
- No debe tener mas de 30 caracteres.
- Sólo se permiten utilizar letras del alfabeto (Inglés), números o signo de subrayado (también el signo $ y #, pero como se utilizan de manera especial no es recomendado).
- No pueden haber dos tablas con el mismo nombre.
- No puede ser una palabra reservada de oracle.
- Las mayúsculas y minúsculas son indiferentes.

## 4.1 RESTRICCIONES

CLAVE PRIMARIA: PRIMARY KEY

- Sólo hay una por tabla
- No puede ser nula
- Puede ser una columna o un conjunto de columnas que identifican unívocamente a una fila.
- Se crea un índice automáticamente

Ejemplo 1: **create table** proveedor (

cod_prov number (2) **primary key**,
nom_prov varchar2(15) ,
dir_prov varchar2(30) );

Ejemplo 2: **create table** proveedor (

cod_prov number (2) **constraint pk_prov primary key,**
nom_prov varchar2(15) ,
dir_prov varchar2(30) );

Ejemplo 3: **create table** proveedor (

cod_prov number (2) ,
nom_prov varchar2(15) ,
dir_prov varchar2(30),
**constraint pk_prov primary key (cod_prov, nom_prov)** );

CLAVE AJENA: FOREIGN KEY...REFERENCES

- Puede ser una o varias columnas que están asociadas a una clave primaria de otra o de la misma tabla
- Pueden definirse tantas claves ajenas como sea preciso
- Ha de cumplir la regla de integridad referencial

Ejemplo 1:

**create table** provincias (
cod_prov number (2) **constraint pk_prov primary key,**
nom_prov varchar2(15) ,
dir_prov varchar2(30) );

Sin poner nombre a la restricción:

**create table** empleado (

nombre varchar2(25) **primary key**,

edad number,

cod_provincia number(2) **references provincias** on delete cascade );

Poniendo nombre a la restricción:

**create table** empleado (

nombre varchar2(25) **constraint pk_emp primary key**,

edad number,

cod_provincia number(2) **constraint fk_em_prov references** provincias on delete cascade );

Ejemplo 2:

**create table** provincias (

cod_prov number (2),

nom_prov varchar2(15) ,

dir_prov varchar2(30),

**constraint pk_prov primary key (cod_prov, nom_prov)** );

Sin poner nombre a la restricción:

**create table** empleado (

nombre varchar2(25) **primary key**,

edad number,

cod_provincia number(2),

nombre_prov varchar2(15),

**foreign key (cod_provincia, nombre_prov) references provincias** on delete cascade );

Poniendo nombre a la restricción:

**create table** empleado (

nombre varchar2(25) **constraint pk_emp primary key**,

edad number,

cod_provincia number(2),

nombre_prov varchar2(15),

**constraint fk_emp_prov foreign key (cod_provincia, nombre_prov) references provincias** on delete cascade );

## OBLIGATORIEDAD: NOT NULL

Una restricción NOT NULL prohíbe que un valor de base de datos sea nulo.

Ejemplo:

```
create table ejemplo (
 nif varchar2(30) not null,
 nombre varchar2 (30) constraint nom_nulo not null,
 edad number(2)    );
```

Nota : En este caso es indiferente ponerle nombre a la restricción not null, porque en caso de error a la hora de dejar el valor de ese campo en blanco, la base de datos siempre te dirá que ese valor no se puede dejar a nulo

## VALORES POR DEFECTO: DEFAULT

- La restricción DEFAULT se utiliza para establecer un valor predeterminado para una columna.
- Es posible incluir varias expresiones: constantes, funciones SQL y variables como UID y SYSDATE
- No se puede hacer referencias a otras columnas o funciones PL/SQL

Ejemplo:

```
create table ejemplo (
 nif varchar2(30) not null,
 nombre varchar2 (30) default 'antonio machado' not null,
 edad number(2),
 fecha date default sysdate );
```

## VERIFICACIÓN DE CONDICIONES: CHECK

- Esta restricción expresa una condición que ha de cumplirse para todas y cada una de las filas de la tabla.
- Puede hacer referencia a una o varias columnas, pero no a valores de otras filas.
- No puede incluir subconsultas ni SYSDATE, UID

Ejemplo:

```
create table ejemplo (
 nif varchar2(10) constraint pk_nif primary key ,
 nombre varchar2(30) not null,
 edad number(2) check (edad between 5 and 20),
 curso number,
 constraint nom_mayus check (nombre=upper(nombre)),
 constraint curso check (curso in(1,2,3))   );
```

UNICIDAD: UNIQUE

- La restricción UNIQUE evita valores repetidos en la misma columna. Puede afectar una o varias columnas.
- Pueden definirse varias columnas con esta restricción
- Admite valores null
- Se crea un índice automáticamente.

Ejemplo 1:

```
create table ejemplo (
 dni varchar2(10) primary key,
 nombre varchar2(30) unique,
 edad number(2)  );
```

Ejemplo 2:

```
create table ejemplo1 (
  dni varchar2(10) primary key,
  nombre varchar2(30),
  edad number(2),
  constraint unica unique(nombre,edad)  );
```

## 5.- BORRADO DE TABLA. DROP

```
DROP  TABLE [usuario.]nombretabla [CASCADE CONSTRAINTS]
```

Cuando se borra la tabla se suprimen también los índices y privilegios asociados a ellas, pero siguen existiendo las vistas y los sinónimos de la tabla aunque dejan de funcionar.

Ejemplo 1:

```
drop table empleado
```

Ejemplo 2:

```
drop table departamento cascade constraints
```

## 6.- MODIFICACIÓN DE ESTRUCTURA DE TABLA. ALTER

Sirve para cambiar la estructura de una tabla, se puede cambiar tanto columnas como restricciones:

```
ALTER TABLE nombretabla

{[ADD  (columna [,columna] ...)]
[MODIFY (columna [,columna] ...)]
[DROP (columna [,columna] ...)]
[ADD CONSTRAINT  restricción]
[DROP CONSTRAINT restricción]};
```

EJEMPLOS:

```
ALTER TABLE empleado ADD
   ( sexo char(1),
     codigo number(4) );
ALTER TABLE empleado MODIFY  ( sexo char (1) NOT NULL );
ALTER TABLE empleado ADD CONSTRAINTS codigo_uq UNIQUE (codigo);
```

## 7.- RENOMBRAR TABLA. RENAME

Permite cambiar una tabla de nombre.

Si hay una clave ajena o vista que hace referencia a la tabla, el cambio de nombre generarán un error.

```
RENAME nombreanterior TO nombre nuevo
```

Las restricciones de integridad, los índices y permisos dados al objeto anterior se transfieren automáticamente al objeto nuevo

Se invalidan los objetos que dependen del objeto renombrado

## 8.- INSERCIÓN DE DATOS. INSERT

```
INSERT INTO tabla [(columna[, columna[, columna]... )]
VALUES (valor[, valor[, valor]...);
```

- Columna: representa las columnas donde se van introducir valores

  Si no se especifican, se consideran, por defecto, todas las columnas
- Valor: representa los valores que se dan a cada una de las columnas
- La asociación valor-columna es posicional
- Los valores tipo carácter y fecha deben ir encerrados entre comillas simples

## 9.- ELIMINACIÓN DE FILAS DE TABLA. TRUNCATE

TRUNCATE TABLE [usuario.]nombretabla [{DROP | REUSE} STORAGE

REUSE STORAGE mantiene reservado el espacio para nuevas filas

Elimina todas las filas de una tabla y libera el espacio ocupado para otros usuarios. No admite retroceso (ROLLBACK).

No se puede truncar una tabla cuya clave primaria sea referenciada por la clave ajena de otra tabla

Ejemplo:

truncate table empleado;

truncate table empleado reuse storage;

## 10.- CONSULTAS. SELECT

La clausula select permite consultar los datos de una o varias tablas según una condición específica.

```
SELECT [ALL | DISTINC] {* | colum1, colum2,..}
FROM {nombretabla1, nombretabla2,...}
[WHERE condición]
[ORDER BY [colum1[ASC|DESC][,column2 [ASC|DESC],...]
```

- Se puede poner un alias a las columnas para su presentación

  Ejemplo: select nom_dep “departamento” from dep:

- Se puede asociar un nombre a las tablas

  Ejemplo: select d.nombre from dep d;

- DISTINCT elimina los valores duplicados (ALL es la opción por omisión)
- ORDER BY especifica el criterio de clasificación del resultado de la columna

  ASC: ordenación ascendente (por defecto)

  DES: ordenación descendente

## 10.1 CLAÚSULA WHERE

Esta claúsula especifica la condición que han de cumplir todas las filas. Las columnas que intervienen en la condición no tienen porqué estar presente en la cláusula select.

Las cadenas de textos son sensibles a mayúsculas y minúsculas y tienen que estar entrecomilladas con comillas simples, sin embargo los números se expresan sin comillas. Para las fechas se debe tener en cuenta el formato de la sesión y se expresa entre comillas simples.

Ejemplo:

Select nombre, apellido from profesores where funcion = ‘Tutor’;

Select nombre, apellido from profesores where codigo = 23;

Select nombre, apellido from profesores where f_ingreso = ‘19/07/2020’

Para la condición de comparación se usan:

- los operadores relacionales <, >, <=, >=, <>, !=, =, in, not in, between, not between
- los operadores ariméticos: +, -, * y /
- Comparación de Fechas, Caracteres, Números
- Caracteres: UPPER, LOWER, ‘like’
- Lógicos: NOT, AND y OR.

Si no existe cláusula WHERE y en la cláusula FROM hay una tabla se recuperan todas las tuplas

Si no existe cláusula WHERE y en la cláusula FROM hay varias tablas, se obtiene el Producto Cartesiano

## 10.2 SUBCONSULTAS CON SELECT

Se puede emplear una sentencia select como parte de otra sentencia select

Sintaxis:

SELECT ... FROM ... WHERE columna operador ( SELECT ...)

Ejemplo:

```
Select nombre, apellido from emple where cod_dep =
        (Select codigo from  departamento where nombre=‘Informatica’);
```

La subconsulta debe estar agrupada por paréntesis

Los operadores que se pueden usar en las consultas son:

= la subconsulta devuelve un valor

IN la subconsulta devuelve una lista de valores

Se puede anidar tantas subconsultas como sean necesarias

## 11.- COMBINACIÓN DE TABLAS CON SELECT

Se especifica en la cláusula where de la siguiente forma

Where tabla1.columna = tabla2.columna:

- Es posible unir tantas tablas como deseemos
- En la cláusula select se pueden especificar columnas de todas las tablas

  Si hay columnas con el mismo nombre en las distintas tablas hay que especificarla de la forma Tabla.Columna
- El criterio de combinación puede estar formado por más de una pareja de columnas

Ejemplo:

```
SELEC t.codigo, desc_tarea, e.dni, nom_emp
FROM tareas t, empleado e
Where t.codigo=e.dni;
```

## 12.- CONSULTAS COMPLEJAS CON SELECT

Sintaxis:

```
SELECT ....FROM …. WHERE ...
GROUP BY columna1, columna2, columna3,….
HAVING condición del
ORDER BY…
```

La Cláusula GROUP BY seguida de una lista de atributos permite agrupar las filas en grupos que tengan los mismos valores en todos los atributos de esa lista.

En una consulta con GROUP BY todos los elementos del select deben ser:

- Expresiones de la cláusula GROUP BY,
- Expresiones con funciones de grupo (sum, avg, ...) o
- Constantes.

Ejemplo: Para cada departamento indica el salario medio

```
SELECT Dpto,  AVG(Salario) FROM Empleado GROUP BY Dpto;
```

La Cláusula HAVING es similar a where, pero trabaja con grupos de filas

## 13.- OPERACIONES DE CONJUNTOS

Las operaciones de conjuntos combinan los resultados de dos o más consultas "select" en un único resultado. Se usan cuando los datos que se quieren obtener pertenecen a distintas tablas y no se puede acceder a ellos con una sola consulta.

Es necesario que las tablas referenciadas tengan tipos de datos similares, la misma cantidad de campos y el mismo orden de campos en la lista de selección de cada consulta.

Sintaxis:

CONSULTA 1

Operador de conjunto

CONSULTA 2

Los operadores de conjuntos son:

- OUTER JOIN (+) : Permite seleccionar los resultados de una tabla aunque no tenga correspondencia con otra .
- UNION : unión SIN duplicados de las filas de las consultas.
- UNION ALL : Unión CON duplicados.
- INTERSECT: Intersección de las filas de las consultas
- MINUS: Diferencia (resta) de conjuntos.

## 14.- CREACIÓN DE TABLAS A PARTIR DE UNA CONSULTA

Con esta sentencia se crea una tabla que contiene datos obtenidos de la consulta.

SINTAXIS:

CREATE TABLE NOMBRE_TABLA [(columna1[,columna2]…]

AS CLAUSULA_SELECT

La consulta puede contener cualquier sentencia select válida (combinación de tabla, subconsulta, etc)

Sólo se copian las restricciones que carecen de nombre y se les asigna un nombre distinto.

Ejemplo:

CREATE TABLE EMPLEYDEPART AS

SELECT E.APELLIDO, D.NOMBRE FROM EMPLE E, DEPAR D

WHERE E.DEPT=D.DEPT

## 15.- INSERCIÓN A PARTIR DE UNA CONSULTA

Podemos insertar varias filas a la vez en una tabla obtenido a partir de una consulta.

SINTAXIS:

```
INSERT INTO NombreTabla1 [(columna1 [,columna2]…)]
SELECT {columna [,columna]... | *}
FROM NombreTabla2 [Cláusulas Select];
```

Las columnas que aparezcan en el SELECT deben tener el mismo número, tipo y orden que las de la tabla en la que se inserta.

Ejemplo:

```
INSERT INTO emple30 SELECT * from emple where dept_no=30;
```

## 16.- MODIFICACIÓN DE VALORES DE COLUMNA. UPDATE

La sentencia update permite modificar el valor de una o varias columnas de una tabla.

SINTAXIS:

```
UPDATE nombretabla
SET columna1=valor1, ...., columnan=valorn
[WHERE columna = condición];
```

Si no se especifica la clausula where se modifican todas las filas de la tabla.

La Condición puede ser una clausula select

Se puede incluir una clausula select en SET de la siguiente forma:

SET (columna1, columna2,...) = (select col1, col2,..)

Ejemplo:

```
Update empleado set salario=salario*2 where dept_no =
(select dept_no from depart where dnombre='Personal');
```

## 17.- BORRADO DE FILAS. DELETE

Esta sentencia permite borrar los datos de una tabla según una determinada condicion.

SINTAXIS:

DELETE [FROM] nombretabla WHERE condición

Si no se especifica la clausula where se borran todas las filas de la tabla.

La condición puede ser una cláusula select

Ejemplo:

```
DELETE CENTROS WHERE COD_CENTRO=50

DELETE FROM EMPLEADO WHERE DEP_NO IN
    (SELECT select dept_no from depart where dnombre=‘Personal’);
```