

U5. 01 Introducción a SQL

- SQL (Structured Query Language), lenguaje de consulta estructurado, es un lenguaje surgido de un proyecto de investigación de IBM para el **acceso a bases de datos relacionales**. Actualmente se ha convertido en un estándar de lenguaje de bases de datos, y la mayoría de los sistemas de bases de datos lo soportan, desde sistemas para ordenadores personales, hasta grandes ordenadores.
- Lenguaje de 4ª generación que nos permite definir y manipular los datos almacenados en la Base de Datos
- Llamado anteriormente SEQUEL proviene de un proyecto de investigación de IBM a mediados de los años setenta sobre la base de datos relacional llamada SYSTEM R
- 1979 Oracle Corporation introduce la primera implementación comercializada de SQL
- IBM desarrolló productos herederos del prototipo SYSTEM R como DB2 y SQL/DS
- ANSI (American National Standards Institute) adoptó SQL como lenguaje estandar para sistemas de BD relacionales en 1986.
- 1987 lo adopta ISO (International Standardization Organization).
- Actualmente lo comercializan diversos SGBD (Sistemas gestores de bases de datos)
 - SQL/DS (SQL/Data System) se ejecuta bajo SO DOS y VMS de IBM
 - Database 2 (DB2) se ejecuta bajo MVS
 - ORACLE
 - DBASE IV
 - INFORMIX
- Visual Basic, por ejemplo, incorpora sentencias SQL para acceder a bases de datos relacionales.
- Características principales:
 - Lenguaje para todo tipo de usuarios: administradores, desarrolladores y usuarios normales.
 - Se especifica **qué** se quiere, no **dónde** ni **cómo** buscarlo
 - Lenguaje para consultas, actualizaciones, definición de datos y control.

Tipos de Sentencias en SQL

El lenguaje SQL proporciona un gran repertorio de sentencias que se utilizan en variadas tareas, como consultar datos de la base de datos, crear, actualizar y eliminar objetos, controlar el acceso a la base de datos y a los objetos. Dependiendo de las tareas, podemos clasificar las sentencias SQL en varios tipos:

- **DML : (Data Manipulation Language)**

Manipulación de datos

SELECT	Recupera filas de la base de datos
INSERT	Añade nuevas filas en la la base de datos
DELETE	Suprime filas en la base de datos
UPDATE	Modifica datos de las filas de la base de datos

- **DDL : (Data Definition Language)**

Definición de datos

CREATE	Table, view, index, synonym...	Crear objetos
DROP		Borrar objetos
ALTER		Modificar la definición de un objeto

- **DCL : (Data Control Language)**

Control de accesos, restricciones

GRANT	Concede privilegios de acceso a usuarios
REVOKE	Suprime privilegios de acceso a usuarios

Componentes sintácticos de una sentencia:

Casi todas las sentencias SQL tienen una forma básica. Comienzan por un verbo, que es una palabra clave que describe que hace la sentencia (por ejemplo SELECT, INSERT, CREATE...), a continuación se especifica los datos con los que opera la sentencia y acaban con cláusulas opcionales u obligatorias que especifican los datos con los que se trabaja.

- La información en una base de datos relacional se almacena en **tablas**.
- Una tabla se compone de una serie de **campos** (también llamados **atributos o columnas**), cada uno de ellos de **un tipo de datos determinado**.

- Una vez definida y creada la tabla se podrán cargar datos en ella, insertar filas que contendrán la información que se quiere almacenar en la base de datos.

Ejemplo:

```
DROP TABLE DEPART;

CREATE TABLE DEPART (
  DEPT_NO NUMBER(2) NOT NULL,
  DNOMBRE VARCHAR2(14),
  LOC VARCHAR2(14) );

INSERT INTO DEPART VALUES (10,'CONTABILIDAD','SEVILLA');
INSERT INTO DEPART VALUES (20,'INVESTIGACION','MADRID');
INSERT INTO DEPART VALUES (30,'VENTAS','BARCELONA');
INSERT INTO DEPART VALUES (40,'PRODUCCION','BILBAO');
```

En este ejemplo hemos creado una tabla y la hemos cargado con datos, cuatro filas o registros.

Cada vez que nos conectemos a la base de datos podremos consultar esta información:

SQL> SELECT * FROM DEPART;

DEPT_NO	DNOMBRE	LOC
10	CONTABILIDAD	SEVILLA
20	INVESTIGACION	MADRID
30	VENTAS	BARCELONA
40	PRODUCCION	BILBAO

U5. 02 CONSULTAS BÁSICAS

Para recuperar información de la base de datos, es decir, para realizar consultas a la base de datos utilizaremos la sentencia SELECT.

Sentencia SELECT

SELECT [ALL | DISTINCT]

<nombre_campo> [{,<nombre_campo>}]

FROM <nombre_tabla>|<nombre_vista>

[{,<nombre_tabla>|<nombre_vista>}]

[**WHERE** <condicion> [{ **AND**|**OR** <condicion>}]]

[**GROUP BY** <nombre_campo> [{,<nombre_campo> }]]

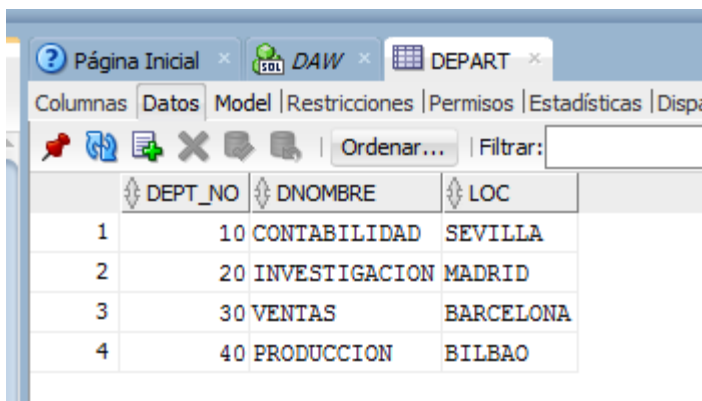
[**HAVING** <condicion>[{ **AND**|**OR** <condicion>}]]

[**ORDER BY** <nombre_campo> [**ASC** | **DESC**]

[{,<nombre_campo> [**ASC** | **DESC** }]]]

La única cláusula obligatoria es la cláusula **FROM** que especifica la tabla o tablas de las que se recuperarán los datos.

Vamos a partir de las siguientes tablas para realizar consultas sobre ellas



	DEPT_NO	DNOBRE	LOC
1	10	CONTABILIDAD	SEVILLA
2	20	INVESTIGACION	MADRID
3	30	VENTAS	BARCELONA
4	40	PRODUCCION	BILBAO

EMP_NO	APELLIDO	OFICIO	RESPONSABLE	FECHA_ALTA	SALARIO	COMISION	DEPT_NO
1	7369 SANCHEZ	ADMINISTRATIVO	7902	17/12/2005	1040	(null)	20
2	7499 ARROYO	VENDEDOR	7698	20/02/2007	1500	390	30
3	7521 SALA	VENDEDOR	7698	22/02/2008	1625	650	30
4	7566 JIMENEZ	DIRECTOR	7839	02/04/2008	2900	(null)	20
5	7654 MARTIN	VENDEDOR	7698	29/09/2012	1600	1020	30
6	7698 ROJO	DIRECTOR	7839	10/05/2012	3005	(null)	30
7	7782 CEREZO	DIRECTOR	7839	09/06/2005	2885	(null)	10
8	7788 GIL	ANALISTA	7566	09/11/2007	3000	(null)	20
9	7839 REY	PRESIDENTE	(null)	17/11/2007	4100	(null)	10
10	7844 TOVAR	VENDEDOR	7698	08/09/2007	1350	0	30
11	7876 ALONSO	ADMINISTRATIVO	7788	23/09/2008	1335	(null)	20
12	7900 JIMENO	ADMINISTRATIVO	7698	03/12/2008	1335	(null)	30
13	7902 FERNANDEZ	ANALISTA	7566	03/12/2007	3000	(null)	20
14	7934 MUÑOZ	ADMINISTRATIVO	7782	23/01/2009	1690	(null)	10

SELECT *

FROM TABLA;

Muestra todas las columnas de la tabla

SELECT campo1, campo2,... campon

FROM TABLA;

Muestra los campos especificados de la tabla.

SQL> select * from depart;

DEPT_NO	DNOMBRE	LOC
10	CONTABILIDAD	SEVILLA
20	INVESTIGACION	MADRID
30	VENTAS	BARCELONA
40	PRODUCCION	BILBAO

SQL> select dept_no, loc from depart;

DEPT_NO	LOC
10	SEVILLA
20	MADRID
30	BARCELONA
40	BILBAO

ALL / DISTINCT

ALL: Recupera todas las filas aunque estén repetidas, es la opción por omisión.

DISTINCT: Sólo recupera las filas que son distintas, no muestra resultados repetidos

```
SELECT ALL DEPT_NO FROM EMPLE;
```

```
SELECT DISTINCT DEPT_NO FROM EMPLE;
```

```
SQL> SELECT DISTINCT DEPT_NO FROM EMPLE;
```

```
DEPT_NO
-----
      10
      20
      30
```

```
SQL> SELECT ALL DEPT_NO FROM EMPLE;
```

```
DEPT_NO
-----
      20
      30
      30
      20
      30
      30
      10
      20
      10
      30
      20
```

```
DEPT_NO
-----
      30
      20
      10
```

14 filas seleccionadas.

Cláusula WHERE

Se utiliza para expresar una o varias condiciones que han de cumplir las filas que se muestran

El formato de la condición es

expresión operador expresión

donde *expresión* puede ser una columna, una expresión aritmética, una constante, un valor nulo o el resultado de aplicar una función sobre cualquiera de los anteriores.

Operadores:

Operadores de comparación:

=, <, >, <=, >=, !=, <>

IN, NOT IN, BETWEEN, NOT BETWEEN, LIKE, NOT LIKE

Operadores lógicos:

OR, AND, NOT

AND: Devuelve TRUE cuando las dos condiciones son verdaderas

OR: Devuelve TRUE cuando al menos una de las dos condiciones es verdadera

NOT: Devuelve TRUE cuando la condición es falsa

Ejemplos:

```
SELECT * FROM EMPLE
```

```
WHERE OFICIO = 'VENDEDOR';
```

```
SELECT * FROM EMPLE WHERE SALARIO > 2500;
```

```
SELECT * FROM EMPLE WHERE SALARIO > 2000 AND SALARIO < 3000;
```

```
SELECT * FROM EMPLE WHERE SALARIO > 2000 OR DEPT_NO <> 30;
```

```
SELECT * FROM EMPLE WHERE SALARIO > 2000 AND DEPT_NO <> 30;
```

```
SELECT * FROM EMPLE
```

```
WHERE OFICIO = 'EMPLEADO' OR DEPT_NO = 30 AND SALARIO < 1600;
```

```
SELECT * FROM EMPLE
```

```
WHERE (OFICIO = 'EMPLEADO' OR DEPT_NO = 30 ) AND SALARIO < 1600;
```

Está permitido utilizar paréntesis para forzar el orden de evaluación.

Comprobación en un conjunto de valores:

- **Operador IN / NOT IN**

Permite comprobar si una columna o expresión pertenece o no a un conjunto de valores

<expresión|columna> IN (conjunto de valores separados por comas)

<expresión|columna> NOT IN (conjunto de valores separados por comas)

```
SELECT * FROM EMPLE
```

```
WHERE DEPT_NO IN (10,30);
```

```
SELECT * FROM EMPLE
```

```
WHERE DEPT_NO NOT IN (10,30);
```

- **Operador BETWEEN / NOT BETWEEN**

Permite comprobar si una columna o expresión está o no dentro de un rango de valores

<expresión|columna> BETWEEN valor_inicial AND valor_final

<expresión|columna> NOT BETWEEN valor_inicial AND valor_final

```
SELECT * FROM EMPLE
```

```
WHERE SALARIO BETWEEN 1800 AND 2500;
```

```
SELECT * FROM EMPLE
```

```
WHERE DEPT_NO NOT BETWEEN 1800 AND 2500;
```

NULL / NOT NULL

Una columna es nula (NULL) cuando está completamente vacía.

Para preguntar si un campo es nulo preguntamos en la condición

CAMPO IS NULL

O para preguntar si no es nulo

CAMPO IS NOT NULL

```
SELECT * FROM EMPLE WHERE COMISION IS NULL;
```

```
SELECT * FROM EMPLE WHERE COMISION IS NOT NULL;
```

Operadores de comparación de cadenas de caracteres: LIKE, NOT LIKE

LIKE nos permite comparar cadenas de caracteres utilizando los siguientes caracteres especiales:

% : Comodín, representa cualquier cadena de 0 o más caracteres

_ : Marcador de posición, representa un carácter cualquiera

```
SELECT * FROM EMPLE WHERE APELLIDO LIKE 'G%';
```


Muestra los empleados cuyo apellido comienza por G

```
SELECT * FROM EMPLE WHERE APELLIDO LIKE '%O%O%';
```

Muestra los empleados cuyo apellido contiene al menos dos O

```
SELECT * EMPLE WHERE APELLIDO LIKE '_A%';
```

Muestra los empleados cuyo apellido tiene una A en la segunda posición

```
SELECT * EMPLE WHERE APELLIDO NOT LIKE '%A';
```

Muestra los empleados cuyo apellido no termina en A

Crear y utilizar alias de columnas

Cuando se consulta la base de datos, los nombres de las columnas se usan como cabeceras de presentación. Si el nombre resulta demasiado largo, corto o críptico, existe la posibilidad de cambiarlo con la misma sentencia SQL de consulta creando un ALIAS. El ALIAS se pone entre comillas dobles a la derecha de la columna.

```
SELECT DNOMBRE "Nombre departamento",  
       DEPT_NO "Número de departamento"  
FROM EMPLE
```

Operadores aritméticos: +, -, *, /

Suma (+) , resta (-) , multiplicación (*) y división (/)

```
SELECT SALARIO, SALARIO + 100 "SALARIO MAS 100" FROM EMPLE
```

```
SELECT SALARIO, SALARIO + 100 SALARIOMAS100 FROM EMPLE;
```

```
SELECT SALARIO + SALARIO*10/100 FROM EMPLE;
```

Cláusula ORDER BY

Sirve para ordenar las filas resultado de la consulta.

```
ORDER BY expre_columna [DESC|ASC], ..., expre_columna [DESC|ASC]
```

Si no se especifica nada el orden es ascendente (ASC)

Esta cláusula es siempre la última dentro de la sentencia SELECT

```
SELECT DEPT_NO, APELLIDO, SALARIO FROM EMPL  
ORDER BY DEPT_NO, APELLIDO;
```

```
SELECT DEPT_NO, APELLIDO , SALARIO FROM EMPL  
ORDER BY SALARIO DESC;
```

```
SELECT DEPT_NO, APELLIDO , SALARIO FROM EMPL  
ORDER BY SALARIO ASC;
```

```
SELECT DEPT_NO, APELLIDO, SALARIO FROM EMPL  
ORDER BY DEPT_NO DESC, APELLIDO ASC;
```