Terminología

SQL

Structured Query Language o Lenguaje de Consultas Estructurado. Es el lenguaje que permite la comunicación con el Sistema Gestor de Bases de Datos (Oracle en nuestro caso).

El SQL es un lenguaje unificado

Lo utilizan todo tipo de usuarios, desde el administrador de la base de datos, DBA, hasta el usuario final.

El SQL es un lenguaje no procedimental.

El usuario especifica Qué quiere, no Cómo ni Dónde conseguirlo.

El SQL es relacionalmente completo.

Permite la realización de cualquier consulta de datos.

SQL= DDL + DML

Las sentencias del SQL se clasifican como parte del DDL o del DML.

Lenguaje de Definición de Datos, DDL

sentencias del SQL que permiten definir los objetos de la Base de Datos (create, revoke, grant, alter, etc.). Cuando se definen dichos objetos se almacenan en el diccionario de datos.

Lenguaje de Manipulación de Datos, DML

sentencias del SQL que se utilizan para manejar los datos de la base de datos (select, insert, update, delete, etc).

commit/rollback

cada vez que se realiza alguna operación en la base de datos no se realiza de forma persistente, sino sobre una copia de la sesión del usuario. Así, si queremos que los resultados de la modificación se trasladen a la base de datos y perduren en el tiempo hay que confirmar dicha transacción con el comando commit. También se puede impedir que los últimos cambios lleguen a efectuarse con rollback, aunque existen algunas sentencias SQL que se 'autoconfirman' y no se pueden volver atrás.

Diccionario de la Base de Datos

Guarda la definición de todos los objetos almacenados en la base de datos; sus características, restricciones, privilegios, relaciones entre ellos, etc.

Tipos de Sentencias

Las sentencias SQL pertenecen a dos categorías principales: Lenguaje de Definición de Datos, DDL y Lenguaje de Manipulación de Datos, DML. Estos dos lenguajes no son lenguajes en sí mismos, sino que es una forma de clasificar las sentencias de lenguaje SQL en función de su cometido. La diferencia principal reside en que el DDL crea objetos en la base de datos y sus efectos se pueden ver en el diccionario de la base de datos; mientras que el DML es el que permite consultar, insertar, modificar y eliminar la información almacenada en los objetos de la base de datos.

Cuando se ejecutan las sentencias DDL de SQL, el SGBD confirma la transacción actual antes y después de cada una de las sentencias DDL. En cambio, las sentencias DML no llevan implícito el commit y se pueden deshacer. Existe pues un problema al mezclar sentencias DML con DDL, ya que estas últimas pueden confirmar las primeras de manera involuntaria e implícita, lo que en ocasiones puede ser un problema.

1.-La Instrucción SELECT COMPLETA

La sintaxis de la instrucción **SELECT** consta básicamente de las cláusulas **SELECT** y **FROM** como obligatorias y de otras varias cláusulas opcionales:

SELECT [ALL | DISTINCT | UNIQUE] <lista-de selecciones>
FROM <nombre de tabla> [alias de tabla] [,...]
[**WHERE** <condición>]
[**GROUP BY** <lista-de columnas> [**HAVING** <condición>]]
[**ORDER BY** <nombre de columna> [**ASC** | **DESC**] [,...]]

SELECT

Permite indicar los datos (columnas o expresiones) que queremos obtener.

SELECT [DISTINCT | UNIQUE] < lista-de selecciones | * >

DISTINCT:	Elimina las filas duplicadas en el resultado de la consulta.
UNIQUE:	Es igual que **DISTINCT**.
<lista-de selecciones>	Lista de nombre de columnas o expresiones separadas por comas.
*	Refiere a todas las columnas de todas las tablas

FROM

Permite indicar las tablas que contienen los datos.

FROM { <nombre de tabla> } [,...]

WHERE

Sirve para indicar la **condición** que deben cumplir las filas resultantes.

WHERE <condición>

*"Una **condición** está formada por una o varias expresiones condicionales conectadas por los operadores lógicos AND, OR y NOT."*

Una **expresión condicional** tiene alguna de las formas siguientes:

- **<expresión1> <operador> <expresión2>**
  Verifica si las dos expresiones satisfacen la comparación.

- **< expresión1> [NOT] BETWEEN <expresión2> AND <expresión2>**
  Verifica si la expresión1tiene un valor comprendido entre los valores de la expresión2 y la expresión3.

- **<expresión> [NOT] IN (<lista-de valores>)**
  Verifica si la expresión tiene un valor de los indicados en la lista de valores.

- **<nombre de columna> [NOT] LIKE "<string>" [ESCAPE "<carácter de escape>"]**
  Verifica si el valor de la columna se adapta al patrón de búsqueda (string). Se admiten caracteres comodín ("%" representa cero o más caracteres, "_" representa un único carácter). El carácter de escape '/' permite referirse a los caracteres comodín como caracteres y no como comodines.

- **<nombre de columna> IS [NOT] NULL**
  Verifica si el valor de la columna es nulo.

Ver tabla de operadores SQL

ORDER BY.

Permite ordenar el resultado de la consulta.

ORDER BY <nombre de columna> [ASC | DESC] [,...]

ASC	los valores de la columna afectada estarán en orden descendente.
DESC	los valores de la columna afectada estarán en orden descendente.

2. Cuestiones de Presentación

Se puede mostrar información calculada a partir de nombres de columnas, números y operadores aritméticos.

Ejemplo: "Obtenga el número de créditos y el nombre de las asignaturas"

```
SELECT creditos, nombre
FROM asignaturas;
```

Ejemplo: "Obtenga el número de horas totales y el nombre de las asignaturas, para ello tenga en cuenta que 1 crédito=10 horas"

```
SELECT creditos*10, nombre
FROM asignaturas;
```

Definición de "alias" de columna

- "Renombra" el nombre de la columna.
- Es útil para expresiones aritméticas.

Ejemplo: "Obtenga el número de horas totales y el nombre de las asignaturas, para ello tenga en cuenta que 1 crédito=10 horas y en la columna correspondiente queremos que aparezca horas_totales"

```
SELECT creditos*10 horas_totales, nombre
FROM asignaturas;
```

Ejemplo: "Obtenga el número de horas totales y el nombre de las asignaturas, para ello tenga en cuenta que 1 crédito=10 horas y en la columna correspondiente queremos que aparezca horas totales"

```
SELECT creditos*10 "horas totales", nombre
FROM asignaturas;
```

Operador de concatenación y literales

- El operador || concatena columnas o cadenas de caracteres a otras columnas.
- Crea una columna resultante que es una expresión CHAR.
- Un literal es un carácter, expresión, o número incluido en la lista del SELECT pero que no es una columna.
- Los literales CHAR y DATE se encierran entre comillas simples, mientras que los números no se encierran entre comillas.

- Para cada tupla de la respuesta se muestra el literal una vez.

Ejemplo:

```
SELECT NOMBRE||'tiene '||  CREDITOS*10
FROM ASIGNATURAS;
```

Ejemplo: “Obtenga los nombres de las asignaturas que tengan entre 3 y 9 créditos’”.

```
SELECT NOMBRE
FROM ASIGNATURAS
WHERE CREDITOS BETWEEN 3 AND 9;
```

```
SELECT NOMBRE
FROM ASIGNATURAS
WHERE CREDITOS IN (3,6,9);
```

Ejemplo: “Obtenga los nombres de las asignaturas que empiecen por A’”.

```
SELECT NOMBRE
FROM ASIGNATURAS
WHERE NOMBRE LIKE 'A%';
```

Ejemplo: “Obtenga el curso y nombre de las asignaturas que empiecen por A y que tengan entre 3 y 9 créditos’”.

```
SELECT NOMBRE, CURSO
FROM ASIGNATURAS
WHERE NOMBRE LIKE 'A%' AND CREDITOS IN (3,6,9);
```

Ordenar

- Se pueden ordenar las filas de una tabla utilizando la cláusula **ORDER BY**.

  - **ASC**: orden ascendente (defecto).
  - **DESC**: orden descendente.

La sintaxis es la siguiente:

```
SELECT [DISTINCT] * | columna [alias] [, columna [alias]…]
FROM tabla [WHERE condicion(es)]
[ORDER BY col [ASC|DESC] [,col [ASC|DESC] …] ;
```

- Se puede ordenar por una columna que no está en la lista de columnas del **SELECT**.
- Se puede ordenar por un alias de una columna.
- Se puede ordenar por más de una columna.

Ejemplo: “Obtenga el curso y nombre de las asignaturas que empiecen por A y que tengan entre 3 y 9 créditos, ordenados de manera ascendente por curso y descendente por nombre’”.

```
SELECT NOMBRE, CURSO
FROM ASIGNATURAS
WHERE NOMBRE LIKE 'A%' AND CREDITOS IN (3,6,9)
ORDER BY CURSO, NOMBRE DESC;
```

Múltiples tablas

- Se pueden seleccionar valores provenientes de múltiples tablas, definiéndolas en la cláusula **FROM** para unirlas a continuación por el/los atributo/s comunes en la cláusula **WHERE**.

```
SELECT tabla1.columna [,tabla1.columna ….], tabla2.columna
[,tabla2. columna ….]
FROM tabla1, tabla2
WHERE tabla1.columna1 = tabla2.columna2
[AND tabla1.columna1 = tabla2.columna2 …..];
```

Ejemplo: “Obtenga el código de docencia, las siglas de la asignatura y la anualidad de la misma”.

```
SELECT D.ID, D.PROFESOR, A.ANUALIDAD
FROM DOCENCIA D, ASIGNATURAS A
WHERE D.SIGLAS=A.SIGLAS;
```