COMPONENTES DEL SQL

Aún cuando se describe a SQL como un lenguaje de consulta, en realidad es mucho más que eso, porque dispone de muchas otras posibilidades además de las de consultar una base de datos. Cada posibilidad tiene su conjunto de instrucciones propias que se expresan:

- Lenguaje de Definición de datos (DDL)
- Lenguaje de Manejo de datos (DML)
- Lenguaje de Control de datos (DCL)
- Lenguaje de Control de Transacciones (TCL)

LDD

TIPOS DE DATOS

Para cada columna tenemos que elegir entre algún dominio definido por el usuario o alguno de los tipos de datos predefinidos que se describen a continuación:

Tipos de datos predefinidos	
Tipos de datos	Descripción
CHARACTER (longitud)	Cadenas de caracteres de longitud fija.
CHARACTER VARYING (longitud)	Cadenas de caracteres de longitud variable.
CHARACTER LARGE OBJECT	Cadena de caracteres de longitud variable hasta el máximo definido por la implementación de la BD
BIT (longitud)	Cadenas de bits de longitud fija.
BOOLEANO (bit(1))	V o F
BINARY LARGE OBJECT	Cadena de bit de longitud variable hasta el max permitido por la implementación. (permite guardar imágenes)
BIT VARYING (longitudb)	Cadenas de bits de longitud variables.
NUMERIC (precisión, escala)	Número decimales con tantos dígitos
DECIMAL (precisión, escala)	Como indique la precisión y tantos decimales como Número decimales con tantos dígitos
INTEGER	Como indique la precisión y tantos decimales como Números enteros.
SMALLINT	Números enteros pequeños.
REAL	Números con coma flotante con precisión predefinida.
FLOAT (precisión)	Números con coma flotante con la precisión especificada.
DOUBLE PRECISION	Números con coma flotante con más precisión predefinida que la del tipo REAL.
DATE	Fechas. Están compuestas de: YEAR año, MONTH mes, DAY día.
TIME	Horas. Están compuestas de HOUR hora, MINUT minutos, SECOND segundos y fraccion de Segundos
TIMESTAMP	Fechas y horas. Están compuestas de YEAR año, MONTH mes, DAY día, HOUR hora, MINUT minutos, SECOND

Restricciones de columna	
Restricción	Descripción
NOT NULL	La columna no puede tener valores nulos.
UNIQUE	La columna no puede tener valores repetidos. Es una clave
PRIMARY KEY	La columna no puede tener valores repetidos ni nulos. Es la clave primaria.
REFERENCES Tabla [(columna)]	La columna es la clave foránea de la columna de la tabla especificada.
CHECK (condiciones)	La columna debe cumplir las condiciones especificadas.

Restricciones de tabla	
Restric	Descripción
UNIQUE (columna [, columna. . .])	El conjunto de las columnas especificadas no puede tener valores repetidos. Es una clave alternativa.
PRIMARY KEY (columna [, columna. . .])	El conjunto de las columnas especificadas no puede tener valores nulos ni repetidos. Es una clave primaria.
FOREIGN KEY (columna [, columna. . .]) REFERENCES tabla [(columna2 [, columna2. . .])]	El conjunto de las columnas especificadas es una clave foránea que referencia la clave primaria formada por el conjunto de las columnas2 de la tabla dada. Si las columnas y las columnas2 se denominan exactamente igual, entonces no sería necesario poner columnas2.
CHECK (condiciones)	La tabla debe cumplir las condiciones especificadas.

LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

Un lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos.

OPERACIONES BÁSICAS DEL DML

Una vez creada la base de datos con sus tablas, debemos poder insertar, modificar y borrar los valores de las filas de las tablas. Para poder hacer esto, el SQL92 nos ofrece las siguientes sentencias:

- **INSERT** para agregar filas a una tabla,
- **UPDATE** para modificar filas de una tabla,
- **DELETE** para borrar filas de una tabla

Una vez que hemos insertado valores en nuestras tablas, tenemos que poder **consultarlos**.
La sentencia para hacer consultas a una base de datos con el SQL92 es:

- **SELECT FROM.**

PALABRAS Y SIMBOLOS PARA LAS CONSULTAS

Las expresiones de valores

Suma	(+)
Resta	(-)
Multiplicación	(*)
División	(%)

Conectores Lógicos

AND OR NOT

Predicados

Un predicado es una condición que se puede evaluar con el fin de que nos dé un valor verdadero que puede ser "verdadero", "falso", o "desconocido". Este resultado se consigue aplicando el predicado a una fila dada de una tabla. Los predicados que se incluyen en SQL son:

Comparación: (=, <>, <, >, <=, =>)

Entre (...BETWEEN...AND...)

IN, (NOT IN)

LIKE

NULL

Cuantificador (ALL, SOME, ANY)

EXISTS, (NOT EXISTS)

BETWEEN. Para expresar una condición que quiere encontrar un valor entre unos límites concretos, podemos utilizar BETWEEN:

```
SELECT nombre_columnas_a_seleccionar
FROM tabla_a_consultar
    WHERE columna BETWEEN límite1 AND límite2;
```

IN. Para comprobar si un valor coincide con los elementos de una lista utilizamos IN, y para ver si no coincide, NOT IN:

```
SELECT nombre_columnas_a_seleccionar
FROM tabla_a_consultar
WHERE columna [NOT] IN (valor1, ..., valorN);
```

LIKE. Para comprobar si una columna de tipo carácter cumple alguna propiedad determinada, podemos usar LIKE:

```
SELECT nombre_columnas_a_seleccionar
FROM tabla_a_consultar
WHERE columna LIKE característica;
```

IS NULL. Para comprobar si un valor es nulo utilizaremos IS NULL, y para averiguar si no lo es, IS NOT NULL.

```
SELECT nombre_columnas_a_seleccionar
FROM tabla_a_consultar
WHERE columna IS [NOT] NULL;
```

ANY/SOME y ALL.

Para ver si un atributo cumple que:

- (ALL) todas sus filas
- (ANY/SOME) algunas de sus filas

satisfacen una condición, podemos hacer:

```
SELECT nombre_columnas_aseleccionar
FROMtabla_a_consultar
WHEREcolumnaoperador_comparación{ALL|ANY|SOME}subconsulta;
```

EXISTS. Para comprobar si una subconsulta produce alguna fila de resultados, podemos utilizar la sentencia denominada test de existencia: EXISTS.
Para comprobar si una subconsulta no produce ninguna fila de resultados, podemos utilizar NOT EXISTS.

```
SELECT nombre_columnas_a_seleccionar
FROM tabla_a_consultar
WHERE [NOT] EXISTS subconsulta;
```

Si se necesita que las filas aparezcan en un orden determinado se deberá utilizar la cláusula **ORDER BY** en la sentencia SELECT

```
SELECT nombre_columnas_a seleccionar
FROM tabla_a_consultar
[WHERE condiciones]
ORDER BY columna_según_la_cual_se_quiere_ordenar [DESC]
[, col_ordenación [DESC]...];
```

- **GROUP BY** nos sirve para agrupar filas según las columnas que indique esta cláusula.

- **HAVING** especifica condiciones de búsqueda para grupos de filas; lleva a cabo la misma función que cumple la cláusula WHERE para las filas de toda la tabla, pero ésta aplica las condiciones a los grupos obtenidos.

```
SELECT nombre_columnas_a seleccionar
FROM tabla_a_consultar
[WHERE condiciones]
GROUP BY
columnas_según_las_cuales_se_quiere_agrupar
[HAVING condiciones_por_grupos]
[ORDERBYcolumna_ordenación[DESC][,columna [DESC]...]];
```

Funciones de agregación	
Función	Descripc
COUNT	Nos da el número total de filas
SUM	Suma los valores de una
MIN	Nos da el valor mínimo de una
MAX	Nos da el valor máximo de una
AVG	Calcula el valor medio de una

- **COUNT (*)** contará todas las filas de la tabla o las tablas que cumplan las condiciones.
- **COUNT (DISTINCT columna),** cuenta las filas que poseen los valores que no fuesen nulos ni repetidos.
- **COUNT (columna),** cuenta los valores que no son nulos en la columna mencionada.

PRODUCTO CARTESIANO

El producto cartesiano crea una sola tabla a partir de las tablas especificadas en la cláusula FROM, haciendo coincidir los valores de las columnas relacionadas de estas tablas.
Si trabajamos con más de una tabla, puede ocurrir que la tabla resultante tenga dos columnas con el mismo nombre. Por ello es obligatorio especificar a qué tabla corresponden las columnas a las que nos estamos refiriendo, denominando la tabla a la que pertenecen antes de ponerlas (por ejemplo, empleados.cuil). Para simplificarlo, se utilizan los alias que, en este caso, se definen en la cláusula FROM.

```
SELECT nombre_columnas_a_seleccionar
FROM tabla1 JOIN tabla2
{ON condiciones|USING (columna [, columna...])}
[WHERE condiciones];
```

La opción **ON**, no solamente exige ser utilizada para expresar condiciones de igualdad, se puede utilizar para expresar condiciones con los demás operadores de comparación que no sean el de igualdad.
También podemos utilizar una misma tabla dos veces con alias diferentes, para distinguirlas.

JOIN INTERNO Y EXTERNO

El JOIN INTERNO (**INNER JOIN**) sólo se queda con las filas que tienen valores idénticos en las columnas de las tablas que compara. Esto puede hacer que perdamos alguna fila interesante de alguna de las dos tablas; por ejemplo, porque se encuentra en NULL en el momento de hacer la combinación.

```
SELECT nombre_columnas_a_seleccionar
FROM t1 [NATURAL] [INNER] JOIN t2
{ON condiciones|
|USING (columna [,columna...])}
[WHERE condiciones];
```

El JOIN EXTERNO (**OUTER JOIN**), nos permite obtener todos los valores de la tabla que hemos puesto a la derecha, los de la tabla que hemos puesto a la izquierda o todos los valores de las dos tablas. Su formato es:

```
SELECT nombre_columnas_a_seleccionar
FROM t1 [NATURAL] [LEFT|RIGHT|FULL] [OUTER] JOIN t2
{ON condiciones|
[USING (columna [,columna...])}
[WHERE condiciones];
```

UNIÓN

La cláusula **UNION** permite unir consultas de dos o más sentencias SELECT FROM. Su formato es:

```
SELECT columnas
FROM tabla
[WHERE condiciones]
UNION [ALL]
SELECT columnas
FROM tabla
[WHERE condiciones];
```

Si ponemos la opción **ALL**, mostrarán las filas obtenidas a causa de la unión. Si no la ponemos, eliminamos las filas repetidas. Lo más importante de la unión es que somos nosotros quienes tenemos que procurar que se efectúe entre columnas definidas sobre **dominios compatibles** (concepto visto en AR). El SQL92 no nos ofrece herramientas para asegurar la compatibilidad semántica entre columnas.

INTERSECCIÓN

Para hacer la intersección entre dos o más sentencias SELECT FROM, podemos utilizar la cláusula INTERSECT, cuyo formato es:

```
SELECT columnas
FROM tabla
[WHERE condiciones]
INTERSECT [ALL]
 SELECT columnas
FROM tabla
[WHERE condiciones];
```

Si indicamos la opción ALL, aparecerán todas las filas obtenidas a partir de la intersección. No la pondremos si queremos eliminar las filas repetidas.

- **Intersección utilizando IN**

```
SELECT columnas
FROM tabla
WHERE columna IN (SELECT columna
FROM tabla
[WHERE condiciones]);
```

- **Intersección utilizando EXISTS**

```
SELECT columnas
FROM tabla
WHERE EXISTS (SELECT *
                FROM tabla
                WHERE condiciones);
```

DIFERENCIA

Para encontrar la diferencia entre dos o más sentencias SELECT FROM podemos utilizar la cláusula EXCEPT, que tiene este formato:

```
SELECT columnas
FROM tabla
[WHERE condiciones]
EXCEPT [ALL] SELECT columnas FROM tabla
[WHERE condiciones];
```

Si ponemos la opción ALL aparecerán todas las filas que da la diferencia. No la pondremos si queremos eliminar las filas repetidas.

- **Diferencia utilizando NOT IN:**

```
SELECT columnas
FROM tabla
WHERE columna NOT IN (SELECT columna
                      FROM tabla
                      [WHERE condiciones]);
```

- **Diferencia utilizando NOT EXISTS:**

```
SELECT columnas
FROM tabla
WHERE NOT EXISTS (SELECT
                  FROM tabla
                  WHERE condiciones);
```