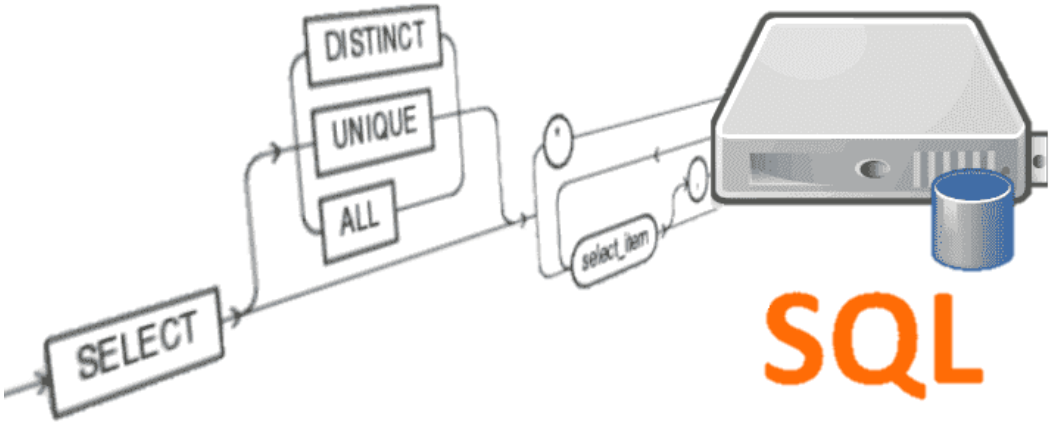
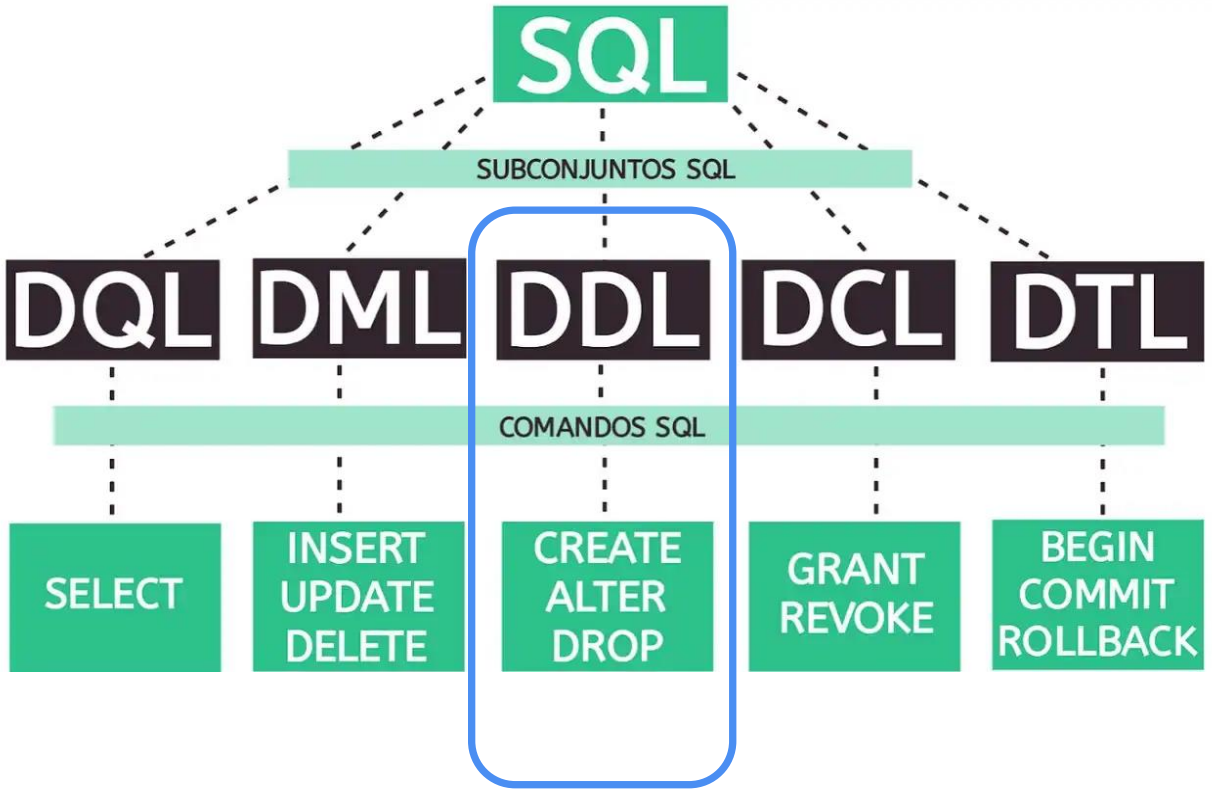
SQL

Structured Query Language

(Lenguaje de Consulta Estructurada)



SQL se divide en Sub Lenguajes



Lenguaje de Definición de Datos

DDL (Data Definition Language):conjunto de comandos para definir, modificar y eliminar la estructura de los objetos de una base de datos.

CREATE: Este comando se utiliza para crear objetos en la base de datos, como tablas, vistas, índices, esquemas y procedimientos almacenados.

CREATE DATABASE nombre_base_datos;

CREATE DATABASE librería

Este comando crea una base de datos llamada librería.

Lenguaje de Definición de Datos

Crear una Tabla

CREATE TABLE nombre_tabla
(nombre_columna1 tipo_dato restriccion,
nombre_columna2 tipo_dato restriccion, ...);

CREATE TABLE empleado (id_empleado **INT PRIMARY KEY**,
nombre **VARCHAR**(50),
sector **VARCHAR**(50),
sueldo **DECIMAL**(10, 2))

Crea una tabla llamada empleado con columnas id_empleado, nombre, sector y sueldo.

Lenguaje de Definición de Datos

```
CREATE TABLE clientes (id_cliente INT PRIMARY KEY,
                       nombre VARCHAR(50) NOT NULL,
                       email VARCHAR(100) UNIQUE,
                       fecha_registro DATE DEFAULT CURRENT_DATE )
```

```
CREATE TABLE pedido(
 id_pedido INT PRIMARY KEY,
 id_cliente INT,
 fecha_pedido DATE,
 monto_total DECIMAL(10, 2),
 CONSTRAINT fk_cliente FOREIGN KEY
 (id_cliente) REFERENCES clientes(id_cliente));
```

Lenguaje de Definición de Datos

ALTER modifica la estructura de un objeto de la base de Datos. Puede agregar, modificar o eliminar columnas en una tabla.

ALTER TABLE Empleados **ADD** fechaNacimiento **DATE**

Este comando añade una columna fechaNacimiento a la tabla empleados

DROP elimina objetos de la base de datos. La eliminación es permanente y no se puede recuperar sin un respaldo

DROP TABLE empleados

Este comando elimina la tabla empleados de la base de datos.

Tipos de datos en SQL

- **INTEGER**: entero binario con signo de palabra completa (31 bits de precisión).
- **SMALLINT**: entero binario con signo de media palabra (15 bits de precisión).
- **DECIMAL (p[,q]**): número decimal con signo de p dígitos de precisión, q decimales. Si q se omite, se asume que vale 0.
- **FLOAT**: numérico con signo de doble palabra y coma flotante.
- **CHAR(n**): cadena de caracteres de longitud fija, de longitud n.
- **VARCHAR(n**): cadena de caracteres de longitud variable, de longitud máxima n.

Las restricciones (constraints)

Son reglas que se aplican a las columnas de una tabla para asegurar la integridad de los datos. Las restricciones pueden aplicarse al crear la tabla o al modificar una tabla existente.

PRIMARY KEY (PK): Define una columna o un conjunto de columnas que identifican de forma única cada fila de la tabla. Una tabla solo puede tener una PRIMARY KEY, y los valores en esta columna no pueden ser nulos ni repetidos

CREATE TABLE empleado (id_empleado **INT PRIMARY KEY**,
nombre **VARCHAR**(50));

Las restricciones (constraints)

FOREIGN KEY (FK): Crea una relación entre dos tablas, conectando la FK de una tabla y la PK de otra. La **integridad referencial** garantiza que el valor FK de una tabla exista en la tabla de referencia como PK

```
CREATE TABLE sector ( id_sector INT PRIMARY KEY,
                      sector VARCHAR(50));
```

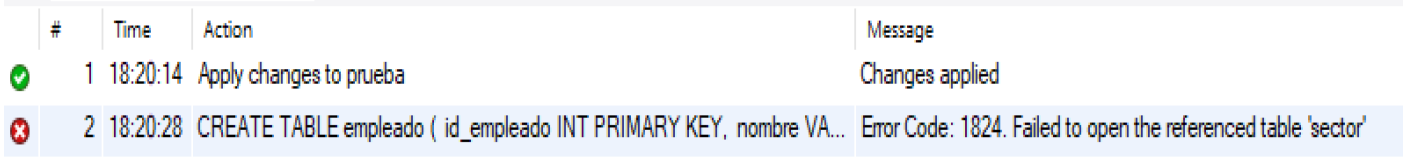
```
CREATE TABLE empleado (
 id_empleado INT PRIMARY KEY,
 nombre VARCHAR(50),
 id_sector INT,
 FOREIGN KEY (id_sector) REFERENCES sector (id_sector));
```

Las restricciones (constraints)

Se debe tener en cuenta que para crear una FK en una tabla, debe existir el atributo clave (PK) en la tabla a la que se quiere hacer referencia, de lo contrario el motor da un error en la creación de la tabla.

```
CREATE TABLE empleado (
 id_empleado INT PRIMARY KEY,
 nombre VARCHAR(50),
 id_sector INT,
 FOREIGN KEY (id_sector) REFERENCES sector (id_sector));

CREATE TABLE sector ( id_sector INT PRIMARY KEY,
                      sector VARCHAR(50));
```



	#	Time	Action	Message
✓	1	18:20:14	Apply changes to prueba	Changes applied
✗	2	18:20:28	CREATE TABLE empleado (id_empleado INT PRIMARY KEY, nombre VA...	Error Code: 1824. Failed to open the referenced table 'sector'

Las restricciones (constraints)

UNIQUE: Asegura que todos los valores de una columna sean únicos. A diferencia de PRIMARY KEY, estas columnas pueden tener valores nulos

NOT NULL: Garantiza que una columna no puede tener valores nulos. Asegura que los campos importantes siempre tengan un valor

CREATE TABLE cliente (id_cliente **INT PRIMARY KEY**,
nombre **VARCHAR**(50) **NOT NULL**,
cuil **VARCHAR**(15) **UNIQUE**);

Las restricciones (constraints)

CHECK: Valida que los datos cumplen con ciertos criterios antes de permitir su inserción o modificación

DEFAULT: Define un valor por defecto para una columna si no se proporciona un valor específico al insertar un registro

CREATE TABLE producto (id_producto **INT PRIMARY KEY**,
nombre VARCHAR(50) **NOT NULL**,
stock **INT DEFAULT** 0,
precio **DECIMAL**(10, 2),
CHECK (precio >0));

Ejemplos - Restricciones

```
CREATE TABLE empleados (
id_empleado INT PRIMARY KEY,  -- asegura unicidad
nombre VARCHAR(50) NOT NULL, -- asegura que el nombre no sea nulo
email VARCHAR(100) UNIQUE,  -- asegura que el email no se repita
sueldo DECIMAL(9, 2) CHECK (sueldo > 0), -- asegura que el sueldo sea positivo
id_depto INT,
FOREIGN KEY (id_depto) REFERENCES
departamentos (id_depto)); -- conecta con otra tabla
```

Ejemplos - Restricciones

```
CREATE TABLE producto ( Id_producto INT,
                        Nombre VARCHAR(30) NOT NULL,
                        Descripcion VARCHAR(255),
                        Precio DECIMAL(9,2),
                        NroProv INT,
CONSTRAINT PK_ producto PRIMARY KEY (Id_producto),
CONSTRAINT FK_Proveedor FOREIGN KEY (NroProv)
REFERENCES Proveedor(Id_proveedor) );
```

El uso de **CONSTRAINT** es opcional en SQL.

Resumen - Restricciones

PRIMARY KEY: Unicidad y no nulo para identificar filas.

FOREIGN KEY: Enlace entre tablas para asegurar integridad referencial.

UNIQUE: Exclusividad en valores no nulos en una columna.

NOT NULL: Evita valores nulos en una columna.

CHECK: Define condiciones específicas para los valores.

DEFAULT: Asigna valores predeterminados en ausencia de datos.

Restricciones que se aplican con el Comando CREATE:

- Clave Primaria **(PRIMARY KEY)**
- Unicidad **(UNIQUE)**
- Obligatoriedad **(NOT NULL)**
- Integridad Referencial **(FOREIGN KEY)**
- Restricciones de Rechazo: Verificación **(CHECK)**

```
CREATE TABLE alumno  ( idAlumno      INT NOT NULL,
                       nombre        VARCHAR(45) NOT NULL,
                       NroSocio      INT,
                       Ingreso       DATE,
                       PRIMARY KEY (idAlumno),
                       UNIQUE INDEX (NroSocio));
```

DDL - ALTER

ALTER modifica la estructura de una tabla. Puede agregar, modificar, o eliminar columnas, y también cambiar las restricciones de la tabla.

Agrega la columna fecha_contratacion de tipo DATE a la tabla empleado

ALTER TABLE empleado **ADD** fecha_contratacion **DATE**

Modifica una columna

Cambia el tipo de dato de la columna sueldo a DECIMAL(15, 2) y agrega la restricción NOT NULL.

ALTER TABLE empleado **MODIFY** sueldo **DECIMAL**(15, 2) **NOT NULL**

DDL - ALTER

Renombrar una columna

ALTER TABLE nombre_tabla
CHANGE COLUMN nom_colum_anterior **TO** nom_colum_nuevo

ALTER TABLE empleado **CHANGE COLUMN** nombre **TO** nomb_completo

cambia el nombre de la columna nombre a nomb_completo

Eliminar una columna

ALTER TABLE empleado **DROP COLUMN** fecha_contratación

Elimina la columna fecha_contratacion de la tabla empleado.

DDL - ALTER

Agregar una restricción

ALTER TABLE empleado **ADD CONSTRAINT** fk_sector **FOREIGN KEY** (id_sector) **REFERENCES** sector (id_sector)

agrega una restricción de clave foránea llamada fk_sector que conecta id_sector de empleado con id_sector de sector

Eliminar una restricción

ALTER TABLE empleado **DROP CONSTRAINT** fk_ fk_sector

Elimina la restricción fk_sector de la tabla empleado.

Las restricciones (constraints)

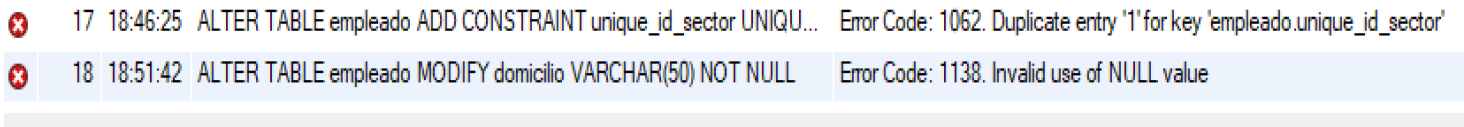
Si se desea agregar restricciones de unicidad o de obligatoriedad en una tabla existente, se debe tener en cuenta que, si la columna elegida no cumple con las restricciones va a dar error.

CREATE TABLE empleado **(** id_empleado **int primary key,**
nombre **varchar(50) not null,** domicilio **varchar(50),** id_sector **int)**

insert into empleado **values** (1,'empleado1',1, default),
(2,'empleado2',1, default), (3,'empleado3',1, default), ,…

Alter table empleado **add constraint** u_id_sector **unique** (id_sector);

Alter table empleado **modify** domicilio **varchar**(50) **not null**;



DDL - ALTER

Renombrar una tabla

ALTER TABLE empleado **RENAME TO** personal

Cliente(id_cliente, cliente, direccion)

Agregar una columna email, agregar una restricción UNIQUE a email, cambiar la el nombre de columna cliente por nombre, eliminar la columna dirección, renombrar la tabla de cliente a usuario

ALTER TABLE cliente **ADD** email **VARCHAR**(100);
ALTER TABLE cliente **ADD CONSTRAINT** unique_email **UNIQUE** (email);
ALTER TABLE cliente **CHANGE COLUMN** cliente **TO** nombre;
ALTER TABLE cliente DROP COLUMN direccion;
ALTER TABLE cliente **RENAME TO** usuario**;**

SELECT INTO

SELECT INTO crea una nueva tabla con los campos que devuelve una consulta SELECT y copia datos en ella. A diferencia de INSERT INTO ... SELECT, SELECT INTO crea y llena la tabla en una sola operación

SELECT columna1, columna2, ...**INTO** nueva_tabla
FROM tabla_existente **WHERE** condición;

columna1, columna2, ...: son las columnas de la **nueva_tabla** (tabla que se va a crear y llenar con los datos seleccionados). Esta tabla no debe existir previamente. **tabla_existente** es la tabla de la que provienen los datos.

SELECT INTO

Crear una copia de una tabla cliente y llamarla cliente_backup
Cliente(id_cliente, nombre, ciudad)

SELECT id_cliente, nombre, ciudad
INTO clientes_backup
FROM cliente

Esta consulta crea la tabla clientes_backup y copia todos los registros de la tabla cliente.

Como quiere hacer una copia exacta de la tabla cliente, se puede usar

SELECT * **INTO** clientes_backup
FROM cliente

SELECT INTO

MySQL no soporta el SELECT INTO, pero se puede realizar la misma tarea con la siguiente consulta

CREATE TABLE nueva_tabla
SELECT columna1, columna2, ...
FROM tabla_existente **WHERE** condición;

columna1, columna2, ...: son las columnas de la **nueva_tabla** (tabla que se va a crear y llenar con los datos seleccionados). Esta tabla no debe existir previamente. **tabla_existente** es la tabla de la que provienen los datos.

SELECT INTO

Tabla con datos filtrados. Crear la tabla cliente_reciente que contenga el id y nombre de los clientes registrado después del 2023-01-01.

Cliente(id_cliente, nombre, fecha_registro)

SELECT id_cliente, nombre **INTO** cliente_reciente
FROM cliente **WHERE** fecha_registro > '2023-01-01’

CREATE TABLE cliente_reciente **SELECT** id_cliente, nombre
FROM cliente **WHERE** fecha_registro > '2023-01-01’

cliente_reciente va a tener solo las dos columnas listadas en el select y todos los registros que satisfagan la condición que las fecha_registro sea posterior a 2023-01-01.

SELECT INTO

Crear una tabla con datos agregados. Muestre el total de ventas por cada cliente en una nueva tabla llamada ventas_totales

Venta(id_cliente, monto)

SELECT id_cliente, **SUM**(monto) AS total_ventas **INTO** ventas_totales
FROM ventas **GROUP BY** id_cliente

CREATE TABLE ventas_totales
SELECT id_cliente, **SUM**(monto) AS total_ventas
FROM ventas **GROUP BY** id_cliente

La tabla ventas_totales tendrá una fila por cada cliente con la suma de sus ventas

SELECT INTO

Crear una tabla combinando datos de varias tablas. crear una tabla empleado_sector que contenga la información de los empleados con el nombre del sector al que pertenece

Empleado(id_empleado, nombre, id_sector)
sector (id_sector, sector)

SELECT e.id_empleado, e.nombre, s.sector **INTO** empleado_sector
FROM empleado e
INNER JOIN sector s **ON** e.id_sector= s.id_sector

CREATE TABLE empleado_sector
SELECT e.id_empleado, e.nombre, s.sector
FROM empleado e **INNER JOIN** sector s **ON** e.id_sector= s.id_sector

Muchas Gracias