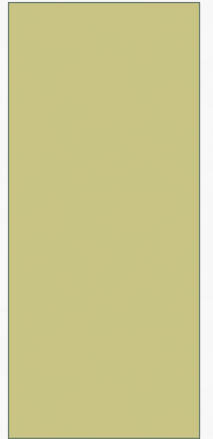


SQL



LENGUAJE SQL BASE DE DATOS



LENGUAJE SQL

- **SQL (Structured Query Language)**, Lenguaje Estructurado de Consulta es el lenguaje utilizado para definir, controlar y acceder a los datos almacenados en una base de datos relacional.
- Como ejemplos de sistemas gestores de bases de datos que utilizan SQL podemos citar DB2, SQL Server, Oracle, MySql, Sybase, PostgreSQL o Access.

LENGUAJE SQL

- El SQL es un lenguaje universal que se emplea en cualquier sistema gestor de bases de datos relacional. Tiene un estándar definido, a partir del cual cada sistema gestor ha desarrollado su versión propia.
- En SQL Server la versión de SQL que se utiliza se llama TRANSACT-SQL.

LENGUAJE SQL

- Las instrucciones SQL se clasifican según su propósito en tres grupos:
 - El **DDL** (Data Description Language) Lenguaje de Descripción de Datos o también conocido como Lenguaje de definición de datos.
 - El **DCL** (Data Control Language) Lenguaje de Control de Datos.
 - El **DML** (Data Manipulation Language) Lenguaje de Manipulación de Datos.

LENGUAJE DCL

- El DCL (Data Control Language) se compone de instrucciones que permiten:
- **Ejercer un control sobre los datos** tal como la asignación de privilegios de acceso a los datos (GRANT/REVOKE).

LENGUAJE DML

- **El DML** se compone de las **instrucciones para el manejo de los datos**, para insertar nuevos datos, modificar datos existentes, para eliminar datos y la más utilizada, para recuperar datos de la base de datos. Veremos que una sola instrucción de recuperación de datos es tan potente que permite recuperar datos de varias tablas a la vez, realizar cálculos sobre estos datos y obtener resúmenes.
- Instrucciones DML:
 - SELECT
 - INSERT
 - UPDATE
 - DELETE

LENGUAJE DDL

- El DDL, es la parte del SQL dedicada a la definición de la base de datos, consta de sentencias para **definir la estructura de la base de datos**, permiten crear la base de datos, crear, modificar o eliminar la estructura de las tablas, crear índices, definir reglas de validación de datos, relaciones entre las tablas, etc.
- Admite las siguientes sentencias de definición:
 - CREATE
 - DROP
 - ALTER

SENTENCIAS DE DEFINICIÓN DE DATOS

- **CREATE.** Con esta instrucción o comando se pueden crear objetos dentro de la base de datos.
- Los objetos que se pueden crear dentro de una base de datos son: tablas, vistas, índices, restricciones, disparadores, procedimientos almacenados o cualquier otro objeto soportado por el motor de base de datos utilizado, incluyendo la base de datos misma.

INSTRUCCIÓN CREATE

- Vamos a examinar la estructura completa de la sentencia CREATE empezando con la más general. Descubrirá que las instrucciones CREATE empiezan de la misma forma y después dan paso a sus especificaciones. La primera parte de CREATE será siempre igual:
- **CREATE <tipo de objeto> <nombre del objeto>**

CREATE DATABASE

- Crea una nueva base de datos y los archivos que se utilizan para almacenar la base de datos
- Sintaxis en su forma mas simple:
- **CREATE DATABASE <nombre de base de datos>**

CREATE DATABASE <nombre de base de datos>

[**ON** [PRIMARY]

([**NAME** = <nombre lógico del archivo> ,]

FILENAME = <'nombre del archivo'>

[, **SIZE** = <tamaño en Kilobytes, megabytes, gigabytes, o terabytes>]

[, **MAXSIZE** = <tamaño en Kilobytes, megabytes, gigabytes, o terabytes>]

[, **FILEGROWTH** = <tamaño en Kilobytes, megabytes, gigabytes, o terabytes |

porcentaje>])]

[**LOG ON**

([**NAME** = <nombre lógico del archivo> ,]

FILENAME = <'nombre del archivo'>

[, **SIZE** = <tamaño en Kilobytes, megabytes, gigabytes, o terabytes>]

[, **MAXSIZE** = <tamaño en Kilobytes, megabytes, gigabytes, o terabytes>]

[, **FILEGROWTH** = <tamaño en Kilobytes, megabytes, gigabytes, o terabytes |

porcentaje>])]

EJEMPLO CREATE DATABASE

```
CREATE DATABASE BD_CLASE
```

```
ON
```

```
( NAME = BD_CLASE_dat,  
  FILENAME = 'C:\Clase\BD_CLASEdat.mdf',  
  SIZE = 5,  
  MAXSIZE = 20,  
  FILEGROWTH = 5 )
```

```
LOG ON
```

```
( NAME =BD_CLASE_log,  
  FILENAME = 'C:\Clase\BD_CLASElog.ldf',  
  SIZE = 2,  
  MAXSIZE = 10,  
  FILEGROWTH = 2 ) ;
```

CREATE TABLE

- Crea una nueva tabla en SQL Server
- Sintaxis:

```
CREATE TABLE nombre_tabla  
(  
  nombre_campo_1 tipo_1  
  nombre_campo_2 tipo_2  
  nombre_campo_n tipo_n  
)
```

EJEMPLO CREATE TABLE

```
CREATE TABLE alumno (  
    carnet INT NOT NULL IDENTITY,  
    nombre VARCHAR(30) NOT NULL,  
    apellido VARCHAR(30) NOT NULL,  
    email VARCHAR(50) NOT NULL ,  
    telefono CHAR(9) NOT NULL,  
    fecha DATETIME,  
    estado SMALLINT NOT NULL,  
    codigo_uni INT NOT NULL  
);
```

EJEMPLO CREATE TABLE CON RESTRICCIONES A NIVEL DE COLUMNA

```
CREATE TABLE alumno (  
    carnet INT NOT NULL IDENTITY PRIMARY KEY,  
    nombre VARCHAR(30) NOT NULL,  
    apellido VARCHAR(30) NOT NULL,  
    email VARCHAR(50) NOT NULL UNIQUE,  
    telefono VARCHAR(9) NOT NULL CHECK (telefono like  
('2[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]')),  
    fecha DATETIME NOT NULL DEFAULT getdate(),  
    estado SMALLINT NOT NULL,  
    codigo_uni INT NOT NULL FOREIGN KEY REFERENCES  
universidad(codigo)  
);
```

EJEMPLO CREATE TABLE CON RESTRICCIONES A NIVEL DE TABLA

```
CREATE TABLE alumno (  
    carnet INT NOT NULL IDENTITY,  
    nombre VARCHAR(30) NOT NULL,  
    apellido VARCHAR(30) NOT NULL,  
    email VARCHAR(50) NOT NULL,  
    telefono VARCHAR(9) NOT NULL,  
    fecha DATETIME NOT NULL,  
    estado SMALLINT NOT NULL,  
    codigo_uni INT NOT NULL  
    CONSTRAINT pk_alumno PRIMARY KEY (carnet),  
    CONSTRAINT u_alumno UNIQUE (email),  
    CONSTRAINT ck_telefono CHECK (telefono like ('2[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]')),  
    CONSTRAINT fk_uni FOREIGN KEY (codigo_uni) REFERENCES  
universidad(codigo)  
);
```


ALTER TABLE

- Modifica una definición de tabla al alterar, agregar o quitar columnas y restricciones, reasignar particiones, o deshabilitar o habilitar restricciones y desencadenadores.
- Algunas instrucciones pueden ser:
- **Agregar un nuevo campo**

```
ALTER TABLE <NOMBRE_TABLA>  
ADD <NOMBRE_CAMPO> <TIPO_DATO> <NULL o NOT NULL>
```

```
ALTER TABLE alumno  
ADD edad INT NOT NULL
```

ALTER TABLE

- **Modificar un campo**

```
ALTER TABLE <NOMBRE_TABLA>  
ALTER COLUMN <NOMBRE_CAMPO> <TIPO_DATO> <NULL o NOT NULL>
```

```
ALTER TABLE alumno  
ALTER COLUMN estado INT NULL
```

- **Agregar una restricción**

```
ALTER TABLE <NOMBRE_TABLA>  
ADD CONSTRAINT <NOMBRE_RESTRICCION> <TIPO_RESTRICCION>  
<VALOR><NOMBRE_CAMPO>
```

```
ALTER TABLE alumno  
ADD CONSTRAINT df_fecha  
DEFAULT getdate() for fecha
```

ALTER TABLE

- **Eliminar un campo**

```
ALTER TABLE <NOMBRE_TABLA>  
DROP COLUMN <NOMBRE_CAMPO>
```

```
ALTER TABLE alumno  
DROP COLUMN edad
```

- **Eliminar una restricción**

```
ALTER TABLE <NOMBRE_TABLA>  
DROP CONSTRAINT <NOMBRE_RESTRICCION>
```

```
ALTER TABLE alumno  
DROP CONSTRAINT df_fecha
```

INSTRUCCIÓN DROP

- **Eliminar una tabla**

DROP TABLE <NOMBRE_TABLA>

DROP TABLE alumno

- **Eliminar una base de datos**

DROP DATABASE <NOMBRE_BASEDEDATOS>

DROP DATABASE BD_CLASE

AL eliminar una base de datos estar seguro de no tenerla en uso, ya que si es así dará error al momento de quererla eliminar