"APRENDIENDO SQL SERVER"

SQL Server es un motor de Base de Datos Gratuito en su versión Express y Developer, que permite almacenar información ordenada y estructurada de manera tal que pueda ser consultada rápidamente. Puede ser utilizado para Crear las bases de datos de varios Sitios Web, Aplicaciones para dispositivos móviles, aplicaciones de Escritorio o bien para almacenar datos de un Videojuego o cualquier aplicación o software empresarial Para Sistemas inventarios y facturas, Sistemas contables, clínicas, hoteles, en fin, todo tipo de estructura de datos para procesamiento, almacenamiento y consultas. Etc.

El SQL es un lenguaje estándar de programación para el acceso a bases de datos, en el existen los lenguajes (DML **Lenguaje de Manipulación de Datos,** DDL **Lenguaje de Definición de Datos,** DCL **Lenguaje de Control de Datos,** TCL **Lenguaje de Control de Transacciones**), los cuales se encargan de administrar como se trabajando con todas sus sentencias, y que luego veremos más adelante, en nuestro caso por ahora solo utilizaremos (DML y DDL), los cuales contienen los comandos necesarios para inicializarnos en el **CRUD** = (**Create, Read, Update, Delete**).

Como he mencionado anteriormente SQL Server, es el entorno que utilizaremos, pero SQL como tal es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.

SQL es un acrónimo en inglés para **Structured Query Language**. Un **Lenguaje de Consulta Estructurado**. Es un tipo de **lenguaje de programación** que te permite manipular y descargar datos de una base de datos. Tiene capacidad de hacer cálculos avanzados y álgebra. Es utilizado en la mayoría de las empresas que almacenan datos en una base de datos. Ha sido y sigue siendo el lenguaje de programación más usado para bases de datos relacionales. Siendo util para acceder y manipular datos en cualquier base de datos del mercado, como, por ejemplo, para las bases de datos MySQL, Oracle, Sybase, Informix, PostgreSQL, DB2, SQL Server, Access, etc.

El SQL se compone de sentencias SQL, cada una con una utilidad diferente, como, por ejemplo:

- Creación de una base de datos (**CREATE DATABASE**)
- Creación de una tabla (**CREATE TABLE**)
- Creación de una vista (**CREATE VIEW**)
- Creación de un índice de una tabla (**CREATE INDEX**)
- Creación de procedimientos almacenados (**CREATE PROCEDURE**)
- Creación de disparadores (**CREATE TRIGGER**)
- Consultar los datos almacenados en una tabla (**SELECT**)
- Insertar datos en una tabla (**INSERT**)
- Modificar datos ya existentes en una tabla (**UPDATE**)
- Borrar datos almacenados en una tabla (**DELETE**)
- Dar permiso de acceso a los datos de una tabla (**GRANT**)
- Eliminar permisos de acceso a datos de una tabla (**REVOKE**)
- Finalizar la transacción de una sentencia SQL (**COMMIT**)
- Retroceder la transacción de una sentencia SQL (**ROLLBACK**).

Bueno, como hemos llegado hasta aquí, pudimos observar que el gestor de Base de Datos **SQL**

Server es uno de los más utilizados por programadores y estudiante de esta área, sus potentes características lo convierten en uno de los sistemas de gestión de Bases de Datos **recomendados** para el desarrollo de software que demandan el manejo de gran cantidad de información.
Una base de datos bien diseñada brinda a los usuarios acceso a información fundamental. Al seguir los principios puedes diseñar una base de datos que funcione bien y se adapte a las necesidades futuras.

1) Comience con el "BIG 6" de las consultas SQL

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

2) Luego, análisis a nivel de grupo con funciones agregadas

COUNT( )
SUM()
AVG( )
MIN( )
MAX( )

3) A continuación, aprenda las relaciones de tabla, el diseño de la base de datos, + tipos de datos

PRIMARY KEY
FOREIGN KEY
CARDINALITY
NORMALIZATION
DATA TYPES

4) Luego, practique la consulta de datos de múltiples tablas

INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL JOIN O UNION

5) A continuación, crea tus propios esquemas + tablas

CREATE SCHEMA
CREATE TABLE
ALTER TABLE
DROP TABLE
INSERT
UPDATE
DELETE

Fuente y Credit: John Pauler

COMANDOS DE AYUDA SQL SERVER.

ALTER TABLE = ATERAR O MODIFCAR UNA TABLA.

```
ALTER TABLE table_name
ADD column_name datatype;
```

ALTER TABLE permite agregar columnas a una tabla de una base de datos.

AND = Y

```
SELECT column_name(s)
FROM table_name
WHERE column_1 = value_1
  AND column_2 = value_2;
```

AND es un operador que combina dos condiciones. Ambas condiciones deben ser verdaderas para que la fila se incluya en el conjunto de resultados.

AS = ALIAS PARA RENOMBRAR COMO

```
SELECT column_name AS 'Alias'
FROM table_name;
```

AS es una palabra clave en SQL que le permite cambiar el nombre de una columna o tabla utilizando un *alias*

AVG() = AVERAGE O PROMEDIO

```
SELECT AVG(column_name)
FROM table_name;
```

AVG() es una función agregada que devuelve el valor medio de una columna numérica.

BETWEEN = ENTRE DOS VALORES

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value_1 AND value_2;
```

El operador BETWEEN se utiliza para filtrar el conjunto de resultados dentro de un cierto rango. Los valores pueden ser números, texto o fechas.

CASE = CASOS

```
SELECT column_name,
  CASE
    WHEN condition THEN 'Result_1'
    WHEN condition THEN 'Result_2'
    ELSE 'Result_3'
  END
FROM table_name;
```

Las instrucciones CASE se utilizan para crear diferentes salidas (normalmente en la instrucción SELECT). Es la forma en que SQL maneja la lógica if-then.

COUNT() = CONTAR

```
SELECT COUNT(column_name)
FROM table_name;
```

COUNT () es una función que toma el nombre de una columna como argumento y cuenta el número de filas donde la columna no es NULL.

.

CREATE TABLE = CREAR UNA TABLA

```
CREATE TABLE table_name (
  column_1 datatype,
  column_2 datatype,
  column_3 datatype
);
```

CREATE TABLE crea una nueva tabla en la base de datos. Le permite especificar el nombre de la tabla y el nombre de cada columna de la tabla.

DELETE = ELIMINAR O BORRAR

```
DELETE FROM table_name
WHERE some_column = some_value;
```

Las instrucciones DELETE se utilizan para quitar filas de una tabla.

GROUP BY

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name;
```

GROUP BY es una cláusula en SQL que solo se usa con funciones agregadas. Se utiliza en colaboración con la instrucción SELECT para organizar datos idénticos en grupos.

HAVING

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY  column_name
HAVING COUNT(*)  >  value;
```

HAVING se agregó a SQL porque la palabra clave WHERE no se podía usar con funciones agregadas COMO sum, max, min, count, avg, etc. Este nos permite poder contar, sumar, avg, max, min, con select y poner una condicion en el having, Es decir, la función HAVING se utiliza para incluir condiciones con alguna función SQL.

INNER JOIN = UNION INTERNA

```
SELECT column_name(s)
FROM table_1
JOIN table_2
   ON table_1.column_name  =  table_2.column_name;
```

Una unión interna combinará filas de diferentes tablas si la *condición de unión* es verdadera.

INSERT = INSERTAR O AGREGAR

```
INSERT INTO table_name  (column_1, column_2, column_3)
VALUES (value_1, 'value_2', value_3);
```

Las instrucciones INSERT se utilizan para agregar una nueva fila a una tabla.

IS NULL / IS NOT NULL

```
SELECT column_name(s)
FROM table_name
WHERE column_name  IS  NULL;
```

IS NULL y IS NOT NULL son operadores utilizados con la cláusula WHERE para probar valores vacíos.

LIKE = BUSCAR PATRON O COINCIDENCIA X

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

LIKE es un operador especial utilizado con la cláusula WHERE para buscar un patrón específico en una columna.

```
SELECT * FROM
WHERE lower(item_name) like '%samsung%' or
      lower(item_name) like '%xiaomi%' or
      lower(item_name) like '%iphone%' or
      lower(item_name) like '%huawei%' or
```

REGEXP_LIKE = BUSCAR DATOS CON VARIAS CONDICIONES.

```
SELECT * FROM table_name
WHERE REGEXP_LIKE(lower(item_name),
'samsung|xiaomi|iphone|huawei')
```

Busca Datos Con Varias Condiciones, es similar a la LIKE condición, excepto que REGEXP_LIKE realiza la coincidencia de expresiones regulares en lugar de la coincidencia de patrones simple realizada por LIKE. Esta condición evalúa cadenas utilizando caracteres definidos por el juego de caracteres de entrada.

LIMIT = LIMITE

```
SELECT column_name(s)
FROM table_name
LIMIT number;
```

LIMIT es una cláusula que le permite especificar el número máximo de filas que tendrá el conjunto de resultados.

MAX() = MAXIMO

```
SELECT MAX(column_name)
FROM table_name;
```

MAX () es una función que toma el nombre de una columna como argumento y devuelve el valor más grande de esa columna.

MIN() = MINIMO

```
SELECT MIN(column_name)
FROM table_name;
```

MIN() es una función que toma el nombre de una columna como argumento y devuelve el valor más pequeño de esa columna.

OR = O

```
SELECT column_name
FROM table_name
WHERE  column_name = value_1
   OR  column_name = value_2;
```

OR = O, es un operador que filtra el conjunto de resultados para incluir solo filas en las que cualquiera de las condiciones es verdadero.

ORDER BY = ORDENAR POR

```
SELECT column_name
FROM table_name
ORDER BY column_name  ASC  |  DESC;
```

ORDER BY es una cláusula que indica que desea ordenar el resultado establecido por una columna en particular, ya sea alfabética o numéricamente.

OUTER JOIN = UNION EXTERIOR

```
SELECT column_name(s)
FROM table_1
LEFT  JOIN table_2
  ON table_1.column_name  =  table_2.column_name;
```

Una unión externa combinará filas de diferentes tablas incluso si no se cumple la condición de unión. Cada fila de la tabla *izquierda* se devuelve en el conjunto de resultados y, si no se cumple la condición de unión, se utilizan valores NULL para rellenar las columnas de la tabla *derecha*.

ROUND() = REDONDEAR A UN VALOR

```
SELECT ROUND(column_name, integer)
FROM table_name;
```

ROUND () es una función que toma un nombre de columna y un entero como argumento. Redondea los valores de la columna al número de decimales especificados por el entero.

SELECT = ESCOGER O SELECCIONAR

```
SELECT column_name
FROM table_name;
```

Las instrucciones SELECT se utilizan para obtener datos de una base de datos. Cada consulta comenzará con SELECT.

SELECT DISTINCT = SELECCIONE DISTINTO

```
SELECT DISTINCT column_name
FROM table_name;
```

SELECT DISTINCT especifica que la instrucción va a ser una consulta que devuelve valores únicos en las columnas especificadas.

SUM = SUMAR

```
SELECT SUM(column_name)
FROM table_name;
```

SUM() es una función que toma el nombre de una columna como argumento y devuelve la suma de todos los valores de esa columna.

UPDATE = ACTUALIZAR O MODIFICAR

```
UPDATE table_name
SET some_column = some_value
WHERE some_column = some_value;
```

Las instrucciones UPDATE permiten editar filas de una tabla, UPDATE = ACTUALIZAR, SET= PONER = WHERE =DONDE.

WHERE = DONDE

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value;
```

WHERE es una cláusula que indica que desea filtrar el conjunto de resultados para incluir solo filas en las que se cumpla la siguiente *condición*.

WITH = CON

```
WITH temporary_name AS (
   SELECT *
   FROM table_name)
SELECT *
FROM temporary_name
WHERE column_name operator value;
```

WITH le permite dar un nombre a un bloque de subconsulta (un proceso también llamado refactorización de subconsulta), al que se puede hacer referencia en varios lugares dentro de la consulta SQL principal. El nombre asignado a la subconsulta se trata como si fuera una vista o tabla en línea. La cláusula SQL WITH es básicamente un reemplazo directo de la subconsulta normal.

Sentencia SQL CREATE VIEW

En SQL, una vista es una tabla virtual basada en el conjunto de resultados de una instrucción SQL. Una vista contiene filas y columnas, como una tabla real. Los campos de una vista son campos de una o más tablas reales de la base de datos. Puede agregar instrucciones y funciones SQL a una vista y presentar los datos como si provinieran de una sola tabla.

```
CREATE VIEW view_name AS
SELECT column1, column2,                                    ...
FROM table_name
WHERE condition;
```

SQL actualizando una vista

Una vista se puede actualizar con la **CREATE OR REPLACE VIEW** declaración.

```
CREATE or alter VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Sintaxis CREAR ÍNDICE

Crea un índice en una tabla. Se permiten valores duplicados:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

CREAR ÍNDICE ÚNICO Sintaxis

Crea un índice único en una tabla. No se permiten valores duplicados:

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

¿Qué es un procedimiento almacenado?

Un procedimiento almacenado es un código SQL preparado que puede guardar, por lo que el código se puede reutilizar una y otra vez.

Entonces, si tiene una consulta SQL que escribe una y otra vez, guárdela como un procedimiento almacenado y luego simplemente llámela para ejecutarla.

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Ejemplo

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
```

Ejecute el procedimiento almacenado anterior de la siguiente manera:

```
EXEC SelectAllCustomers;
```

¿Qué es un Trigger?

Un trigger es una técnica en SQL Server que permite automatizar una acción basada en un evento que ocurre en una tabla específica. Un trigger se activa cuando se ejecuta una operación determinada en la tabla, como una inserción, actualización o eliminación de registros. Un trigger puede ejecutar una serie de sentencias SQL para realizar una acción en respuesta al evento.

```
CREATE TRIGGER nombre_del_trigger
ON nombre_de_la_tabla
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    --código aquí
END
```