ADVISERTECNOLOGY

Learning Innovation Because life is Changing

"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 (**Structured Query Language**) es un lenguaje estándar de programación utilizado para el acceso a bases de datos.

Dentro del SQL existen diferentes tipos de lenguajes, los cuales se encargan de administrar cómo se trabaja con la base de datos:

1) **DML (Data Manipulation Language):** Este lenguaje se utiliza para realizar operaciones de manipulación de datos en una tabla. Algunos ejemplos de estas operaciones son la inserción, actualización y eliminación de registros. El lenguaje DML incluye comandos como *"INSERT", "UPDATE" y "DELETE"*.
2) **DDL (Data Definition Language):** El lenguaje DDL se utiliza para definir la estructura de la base de datos. Entre las tareas que se pueden realizar con este lenguaje se encuentran la creación de tablas, la definición de campos, la especificación de claves primarias y foráneas, la creación de vistas, entre otras. Los comandos más comunes del lenguaje DDL son "*CREATE", "ALTER" y "DROP*".
3) **DCL (Data Control Language):** Este lenguaje se utiliza para controlar el acceso a los datos y establecer permisos de usuario y roles. El lenguaje DCL incluye comandos como "*GRANT"*, que otorga permisos de acceso a los datos, y "*REVOKE*", que revoca dichos permisos.
4) **TCL (Transaction Control Language):** El lenguaje TCL se utiliza para controlar las transacciones. Una transacción es un conjunto de operaciones que se realizan en la base de datos y que deben ser confirmadas o deshechas en conjunto. El lenguaje TCL incluye comandos como *"COMMIT*", que confirma los cambios realizados en la transacción, y "*ROLLBACK"*, que deshace dichos cambios.

Los lenguajes DML y DDL son los más utilizados en SQL, y contienen los comandos necesarios para realizar las operaciones básicas de CRUD (Create, Read, Update, Delete), que son:

1) **Create:** se utiliza para crear una nueva tabla en la base de datos.
2) **Read:** se utiliza para leer datos de una tabla.
3) **Update:** se utiliza para actualizar datos existentes en una tabla.
4) **Delete:** se utiliza para eliminar datos de una tabla.

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:

1) **CREATE TABLE:** este comando se utiliza para crear una nueva tabla en la base de datos, especificando sus campos y tipos de datos.
2) **CREATE VIEW:** este comando se utiliza para crear una vista, que es una tabla virtual que muestra solo los datos específicos que se necesitan, y que se actualiza automáticamente según los datos de la tabla base.
3) **CREATE INDEX:** este comando se utiliza para crear un índice en una tabla, lo que mejora el rendimiento de las consultas al permitir una búsqueda más rápida de los datos.
4) **CREATE PROCEDURE:** este comando se utiliza para crear un procedimiento almacenado, que es un conjunto de instrucciones SQL que se almacenan en la base de datos y se pueden llamar desde otras partes de la aplicación.
5) **CREATE TRIGGER:** este comando se utiliza para crear un disparador, que es un tipo de procedimiento almacenado que se ejecuta automáticamente en respuesta a ciertos eventos en la base de datos, como la inserción, actualización o eliminación de registros en una tabla.
6) **SELECT:** este comando se utiliza para consultar los datos almacenados en una tabla, permitiendo filtrar y ordenar los datos según ciertos criterios.
7) **INSERT:** este comando se utiliza para insertar nuevos datos en una tabla.
8) **UPDATE:** este comando se utiliza para modificar datos ya existentes en una tabla.
9) DELETE: este comando se utiliza para eliminar datos almacenados en una tabla.
10) **GRANT**: este comando se utiliza para dar permiso de acceso a los datos de una tabla a un usuario o grupo de usuarios específico.
11) **REVOKE**: este comando se utiliza para eliminar los permisos de acceso a los datos de una tabla previamente otorgados a un usuario o grupo de usuarios.
12) **COMMIT**: este comando se utiliza para finalizar una transacción de una sentencia SQL, confirmando los cambios realizados en la base de datos.
13) **ROLLBACK**: este comando se utiliza para retroceder la transacción de una sentencia SQL, deshaciendo los cambios realizados en la base de datos y volviendo al estado anterior.

Después de nuestro recorrido por **SQL Server**, podemos afirmar que este gestor de bases de datos es uno de los más populares entre los **desarrolladores, programadores, analistas de datos, y estudiantes de informática.** Sus potentes características lo convierten en una opción recomendada para el desarrollo de software que requiere el manejo de grandes cantidades de información.

Es importante destacar que una base de datos bien diseñada puede ofrecer a los usuarios acceso a información fundamental. Al seguir principios de diseño, es posible crear una base de datos que

funcione de manera óptima y se adapte a las necesidades futuras.

Algunos principios de diseño incluyen la normalización de datos para reducir la redundancia y mejorar la integridad de la información, la definición de claves primarias y foráneas para establecer relaciones entre tablas, y la consideración de la escalabilidad y el rendimiento a largo plazo.

Estos comandos son la base de las consultas SQL y se utilizan para recuperar datos específicos de una base de datos en función de ciertos criterios.

Los "BIG 6" de las consultas SQL son un conjunto de seis comandos esenciales utilizados en las consultas SQL para recuperar información de una base de datos. Estos comandos son:

a) **SELECT**: utilizado para seleccionar las columnas específicas que se desean recuperar de una o más tablas.
b) **FROM**: utilizado para especificar las tablas de donde se recuperarán los datos.
c) **WHERE**: utilizado para filtrar los resultados de la consulta basándose en condiciones específicas.
d) **GROUP BY:** utilizado para agrupar los resultados de la consulta en función de una o más columnas.
e) **HAVING:** utilizado para filtrar los resultados de la consulta basándose en condiciones específicas después de la cláusula GROUP BY.
f) **ORDER BY:** utilizado para ordenar los resultados de la consulta por una o más columnas en orden ascendente o descendente.

Funciones agregadas para análisis a nivel de grupo:

Las funciones agregadas son un conjunto de funciones que se utilizan en SQL para realizar cálculos sobre conjuntos de valores. En el contexto de análisis a nivel de grupo, estas funciones se utilizan para resumir datos en una tabla según los valores de una o varias columnas.

Las funciones agregadas más comunes utilizadas en SQL incluyen:

a) **COUNT():** se utiliza para contar el número de filas en una tabla o el número de valores no nulos en una columna.
b) **SUM():** se utiliza para sumar los valores de una columna.
c) **AVG():** se utiliza para calcular el promedio de los valores de una columna.
d) **MIN():** se utiliza para encontrar el valor mínimo de una columna.
e) **MAX():** se utiliza para encontrar el valor máximo de una columna.

Estas funciones pueden ser utilizadas junto con la cláusula GROUP BY, la cual agrupa los datos por los valores de una o varias columnas y permite realizar cálculos para cada grupo individualmente.

Relaciones de tabla, diseño de la base de datos y tipos de datos:

Las relaciones de tabla, el diseño de la base de datos y los tipos de datos son elementos importantes en el manejo de bases de datos con SQL. A continuación, se explica cada uno de ellos:

1) **Relaciones de tabla:** Las relaciones de tabla establecen la conexión entre dos o más tablas de una base de datos. Por ejemplo, si una tabla "Usuarios" tiene una columna "ID" y otra tabla "Pedidos" tiene una columna "ID_usuario", se puede establecer una relación entre ambas tablas a través de la columna "ID". Esto permite realizar consultas y combinaciones de datos entre las tablas.

2) **Diseño de la base de datos:** El diseño de la base de datos es la estructura general que se va a utilizar para almacenar y organizar la información. Esto incluye la creación de tablas, la definición de campos y la determinación de claves primarias y foráneas. Un buen diseño de la base de datos puede mejorar el rendimiento y la eficiencia del sistema.

3) **Tipos de datos:** Los tipos de datos se refieren al formato de los valores que se almacenan en las tablas de la base de datos. Algunos de los tipos de datos más comunes en SQL incluyen: texto, números enteros, decimales, fecha y hora, entre otros. Es importante elegir el tipo de dato correcto para cada columna de una tabla, ya que esto puede afectar el rendimiento de las consultas y el almacenamiento de datos.

Algunos de los comandos mas utilizados para trabajar con las relaciones:

a) PRIMARY KEY: se utiliza para identificar de manera única cada fila en una tabla.
b) FOREIGN KEY: se utiliza para establecer una relación entre dos tablas.
c) CARDINALIDAD: se refiere a la relación entre el número de filas en una tabla y el número de filas en otra tabla relacionada.
d) NORMALIZACIÓN: se utiliza para eliminar la redundancia de los datos y mejorar la eficiencia del sistema de base de datos.
e) DATA TYPES: se refiere a los tipos de datos que pueden ser almacenados en una columna, como texto, números, fechas, etc.

Consulta de datos de múltiples tablas:

a) INNER JOIN: se utiliza para combinar los datos de dos tablas en función de una clave común.
b) LEFT JOIN: se utiliza para devolver todas las filas de la tabla de la izquierda y las filas coincidentes de la tabla de la derecha.
c) RIGHT JOIN: se utiliza para devolver todas las filas de la tabla de la derecha y las filas coincidentes de la tabla de la izquierda.
d) FULL JOIN o UNION: se utiliza para devolver todas las filas de ambas tablas, incluyendo las filas que no tienen coincidencias en la otra tabla.

Creación de esquemas y tablas:

a) CREATE SCHEMA: se utiliza para crear un esquema para organizar y agrupar objetos de base de datos.
b) CREATE TABLE: se utiliza para crear una nueva tabla en la base de datos.
c) ALTER TABLE: se utiliza para modificar la estructura de una tabla existente.
d) DROP TABLE: se utiliza para eliminar una tabla de la base de datos.
e) INSERT: se utiliza para agregar una nueva fila a una tabla existente.
f) UPDATE: se utiliza para modificar los valores de una o más filas en una tabla existente.
g) DELETE: se utiliza para eliminar una o más filas de una tabla existente.

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
```

Como haz visto en este documento, SQL Server es una poderosa herramienta de administración de bases de datos relacionales con una amplia gama de características, incluyendo sentencias SELECT, INSERT, UPDATE, DELETE, DROP, ALTER, TRUNCATE, entre otras. También cuenta con una variedad de funciones integradas, como AVG, SUM, COUNT, MAX, MIN, GROUP BY, ORDER BY, HAVING, CONCAT, CONVERT, LOWER, UPPER, DATEADD, DATETIME, GETDATE, DATEDIFF, DATEPART, CASE, WHEN, LIKE, ABS, CEIL, FLOOR, ROUND, TRUNC, entre otras.

Además, SQL Server cuenta con consultas avanzadas, como INNER JOIN, LEFT JOIN, RIGHT JOIN, OUTER JOIN, FULL JOIN, CROSS JOIN, etc. Estas consultas avanzadas permiten unir datos de diferentes tablas en una sola consulta, lo que permite una mayor flexibilidad y capacidad de análisis.

Es importante seguir aprendiendo y practicando con todas estas sentencias, funciones y características para aprovechar al máximo las capacidades de SQL Server. Te Recomiendo visitar mi blog https://advisertecnology.com/, el canal de Youtube de Juancito Peña (https://www.youtube.com/@JuancitoPenaV), y seguirme en mis redes sociales para acceder a más material de apoyo, consejos y tips para mejorar tus habilidades en SQL Server. No te detengas, sigue aprendiendo y practicando para convertirte en un experto en SQL Server.