INFORME: SQL Y COMANDOS AVANZADOS

INTRODUCCIÓN A SQL (STRUCTURED QUERY LANGUAGE)

SQL (Structured Query Language) es el lenguaje estándar para interactuar con bases de datos relacionales. Permite realizar diversas operaciones como creación, modificación y eliminación de objetos de bases de datos, así como la manipulación de los datos almacenados en ellas. Se divide en varias categorías principales:

- **DDL (Data Definition Language):** Define y modifica la estructura de la base de datos (tablas, índices, vistas, etc.).
- **DML (Data Manipulation Language):** Manipula los datos dentro de las estructuras definidas (inserción, actualización y eliminación).
- **DQL (Data Query Language):** Se utiliza para realizar consultas y recuperar datos.

1. DDL (DATA DEFINITION LANGUAGE)

1.1 Comando CREATE TABLE

El comando **CREATE TABLE** se utiliza para crear una nueva tabla en la base de datos. Ejemplo:

```
CREATE TABLE Clientes ( ID INT PRIMARY KEY, Nombre VARCHAR(50), Correo VARCHAR(100), FechaRegistro DATE );
```

1.2 Restricciones

Las restricciones garantizan la integridad de los datos en una tabla. Algunas restricciones comunes incluyen:

- **PRIMARY KEY:** Identifica de manera única cada registro.
- **FOREIGN KEY:** Crea relaciones entre tablas.
- **NOT NULL**: Impide valores nulos en una columna.
- **UNIQUE:** Garantiza que todos los valores en una columna sean únicos.
- **CHECK:** Define condiciones que los datos deben cumplir.
- **DEFAULT**: Asigna un valor por defecto si no se proporciona uno.

1.3 Tipos de Datos

SQL Server admite una variedad de tipos de datos, incluyendo:

- **INT:** Números enteros.
- **VARCHAR(n):** Cadenas de texto de longitud variable.
- **DECIMAL(p,s):** Números con decimales.
- **DATE:** Fechas.
- **BIT:** Valores booleanos (0 o 1).

1.4 Comandos para Índices

CREATE INDEX: Crea un índice para mejorar el rendimiento de las consultas.

```
CREATE INDEX idx_nombre ON Clientes(Nombre);
```

- DROP INDEX: Elimina un índice existente.

```
DROP INDEX idx_nombre ON Clientes;
```

1.5 Modificación de Tablas

- **ALTER TABLE:** Permite modificar la estructura de una tabla existente.

```
ALTER TABLE Clientes ADD Telefono VARCHAR(20);
```

- **DROP TABLE:** Elimina una tabla de la base de datos.

```
DROP TABLE Clientes;
```

1.6 Vistas

- **CREATE VIEW:** Crea una vista virtual basada en una consulta.

```
CREATE VIEW VistaClientes AS SELECT Nombre, Correo FROM Clientes WHERE FechaRegistro >= '2023-01-01';
```

- **ALTER VIEW:** Modifica una vista existente.

```
ALTER VIEW VistaClientes AS SELECT Nombre, Telefono FROM Clientes;
```

- **DROP VIEW:** Elimina una vista.

```
DROP VIEW VistaClientes;
```

1.7 Triggers (Disparadores)

Los triggers son procedimientos almacenados que se ejecutan automáticamente en respuesta a eventos en una tabla.

- **CREATE TRIGGER:** Crea un trigger.

```
CREATE TRIGGER trg_AntesInsert ON Clientes BEFORE INSERT AS BEGIN PRINT 'Se está insertando un nuevo cliente'; END;
```

- **ALTER TRIGGER:** Modifica un trigger existente.
- **DROP TRIGGER:** Elimina un trigger.

2. DML (DATA MANIPULATION LANGUAGE)

2.1 Comando GROUP BY y HAVING

- **GROUP BY:** Agrupa filas que tienen valores comunes.
- **HAVING:** Filtra grupos de datos.

```
SELECT Categoria, COUNT(*) AS Total FROM Productos GROUP BY Categoria HAVING COUNT(*) > 10;
```

2.2 Funciones de Agrupación

- **SUM:** Calcula la suma de valores.
- **MAX**: Devuelve el valor máximo.
- **MIN**: Devuelve el valor mínimo.
- **AVG:** Calcula el promedio.
- **COUNT**: Cuenta el número de registros.

3. FUNCIONES EN SQL

3.1 Funciones de Fecha

- **GETDATE():** Devuelve la fecha y hora actual.
- **DATEDIFF():** Calcula la diferencia entre dos fechas.
- **DATEPART():** Devuelve una parte específica de una fecha.

```
SELECT GETDATE(), DATEDIFF(day, '2024-01-01', GETDATE()), DATEPART(month, GETDATE());
```

3.2 Funciones de Texto

- **SUBSTRING():** Extrae una parte de una cadena.
- **LEFT() / RIGHT():** Devuelve los caracteres desde la izquierda o derecha.
- **UPPER():** Convierte a mayúsculas.
- **RTRIM() / LTRIM():** Elimina espacios en blanco a la derecha/izquierda.
- **REPLACE():** Reemplaza partes de una cadena

```
SELECT UPPER(Nombre), SUBSTRING(Nombre, 1, 3) FROM Clientes;
```

3.3 Funciones de Sistema

- **ISNULL():** Reemplaza valores nulos por un valor especificado.
- **COALESCE():** Devuelve el primer valor no nulo de una lista.
- **DATALENGTH():** Devuelve la longitud en bytes de un valor.

4. CONSULTAS AVANZADAS Y MANIPULACIÓN DE DATOS

4.1 Comando UPDATE

- Modifica registros existentes en una tabla.

```
UPDATE Clientes SET Correo = 'nuevoemail@example.com' WHERE ID = 1;
```

4.2 Comando DELETE

- Elimina registros específicos de una tabla.

```
DELETE FROM Clientes WHERE ID = 1;
```

4.3 Comando TRUNCATE TABLE

- Elimina todos los registros de una tabla, pero conserva la estructura.

```
TRUNCATE TABLE Clientes;
```

CONCLUSIÓN

SQL es el pilar fundamental en el manejo de bases de datos relacionales, ya que permite crear y gestionar estructuras, manipular datos y realizar consultas avanzadas de manera eficiente. A lo largo del tiempo, SQL ha evolucionado para soportar no solo operaciones básicas, sino también funcionalidades complejas que aseguran la integridad de los datos y mejoran el rendimiento de las aplicaciones. Comandos como **CREATE TABLE** y **ALTER TABLE** permiten definir y modificar la estructura de las bases de datos, mientras que el uso de restricciones como **PRIMARY KEY** y **FOREIGN KEY** garantiza que los datos cumplan con reglas específicas, manteniendo su consistencia.

Por otro lado, el manejo de datos a través de **DML (Data Manipulation Language),** con comandos **como INSERT, UPDATE y DELETE**, facilita la manipulación eficiente de grandes volúmenes de información. Las funciones de agrupación, como **SUM, AVG y COUNT**, son útiles para obtener resúmenes de datos, mientras que las funciones de texto y fecha permiten transformar y analizar los datos en distintos formatos. Además, las herramientas avanzadas como los triggers y las vistas ayudan a automatizar procesos y mejorar el acceso a la información, respectivamente.

La capacidad de realizar consultas avanzadas con cláusulas como **GROUP BY** y **HAVING** permite extraer información clave de los datos almacenados, lo que es esencial para la toma de decisiones informadas. Asimismo, el uso de funciones del sistema, como **ISNULL** y **COALESCE**, contribuye a gestionar valores nulos y asegurar resultados consistentes en las consultas.

En resumen, el conocimiento profundo de SQL no solo es indispensable para los administradores de bases de datos, sino también para los desarrolladores y analistas que buscan optimizar sus aplicaciones y extraer valor de la información. SQL sigue siendo una herramienta poderosa y versátil en el entorno tecnológico actual, y su dominio proporciona una ventaja significativa en el manejo de datos de manera segura, eficiente y flexible.

.