Una de las claves en la eficiencia de manejo de información, es la capacidad de utilizar las herramientas más útiles para acelerar procesos y garantizar un rendimiento óptimo en la manipulación de datos.

A continuación te presentamos algunas que te permitirán, a través del uso de tabla temporales, simplificar tu flujo de trabajo.

Cláusula Table

TABLE es una cláusula que funciona como un SELECT simplificado y más limitado, pero que permite ser más conciso.

- Retorna las filas y columnas de una tabla específica.
- Puedes usar la cláusula ORDER BY para ordenar los resultados y LIMIT para limitar el número de filas devueltas.

```
TABLE nombre_tabla [ORDER BY columna] [LIMIT numero]
```

TABLE difiere de SELECT en dos aspectos clave:

- TABLE siempre muestra todas las columnas de la tabla.
- TABLE no permite ningún filtrado arbitrario de filas; es decir, TABLE no admite ninguna cláusula WHERE.

Cláusula VALUES

La declaración **VALUES** permite crear una tabla temporal con valores específicos. Esta tabla temporal es especialmente útil cuando necesitas realizar operaciones en un conjunto de datos que no proviene de una tabla real.

Estructura de la declaración VALUES:

```
VALUES ROW(valor1, valor2), ROW(valor1, valor2), ROW(valor1, valor2);
```

La sentencia se inicia con la palabra clave VALUES, a continuación se especifica una lista de uno o más constructores de filas, separados por comas, cada constructor de fila comienza con ROW(), seguido de una lista de valores entre paréntesis.

Los valores dentro de un constructor de fila pueden ser de diferentes tipos de datos.

Ejemplo:

```
VALUES ROW('Alice', 25, 'New York'), ROW('Bob', 30, 'Los Angeles'),
ROW('Charlie', 22, 'Chicago');
```

Esto generará una tabla temporal con tres filas, donde cada fila contiene información sobre una persona.

Si quisiéramos hacer operaciones de filtrado sobre esta tabla temporal deberíamos hacer lo siguiente:

```
SELECT * FROM ( VALUES ROW('Alice', 25, 'New York'), ROW('Bob', 30, 'Los
Angeles'), ROW('Charlie', 22, 'Chicago') ) as mi_tabla_temporal WHERE
condiciones;
```

Como verás es incomodo operar de esta manera, por eso en el apartado siguiente veremos cómo crear tablas temporales que puedan referenciarse más fácilmente.

Tablas temporales

Las tablas temporales son tablas que existen durante la duración de una sesión de MySQL y se eliminan automáticamente al finalizar la sesión. Tiene la misma sintaxis que el CREATE TABLE que ya hemos aprendido, la única diferencia es la cláusula TEMPORARY.

```
CREATE TEMPORARY TABLE nombre_tabla_Temporal(
    Id INT,
    nombre VARCHAR(50),
    edad INT
);
```

La ventaja de usar tablas temporales, es que se pueden crear a partir de una cláusula SELECT o VALUES tambien, permitiendo tener un nombre que las referencie en lugar de tener que usar la consulta completa todo el tiempo:

```
CREATE TEMPORARY TABLE mayores_de_25 AS
SELECT nombre, edad FROM nombre_tabla WHERE edad > 25;

#ahora podemos referenciar la consulta de una manera más simple
SELECT * FROM mayores_de_25;
TABLE mayores_de_25;
```

Ejemplo con SELECT:Ejemplo con VALUES:

```
CREATE TEMPORARY TABLE tabla_temporal AS
VALUES ROW(1, 'Alice', 25), ROW(2, 'Bob', 30), ROW(3, 'Charlie', 22);

#ahora podemos referenciar la consulta de una manera más simple
SELECT * FROM tabla_temporal;
# o
TABLE tabla_temporal;
```

💡*Si necesitas probar la creación de una tabla temporal muchas veces puedes usar el comando 'DROP TEMPORARY TABLE IF EXISTS nombre_tabla_temporal'. De esta manera al ejecutar ambas querys nunca te saltara el error de que ya existe la tabla temporal:*

```
DROP TEMPORARY TABLE IF EXISTS tabla_temporal;
CREATE TEMPORARY TABLE tabla_temporal AS
VALUES ROW(1, 'Alice', 25), ROW(2, 'Bob', 30), ROW(3, 'Charlie', 22);
```

Introducción a las Operaciones de Conjunto (set operations)

Las operaciones de conjunto en SQL combinan los resultados de múltiples bloques de consulta en un solo resultado. Un bloque de consulta, es cualquier declaración SQL que devuelve un conjunto de resultados, como SELECT, pero también admite las declaraciones TABLE y VALUES.

El estándar SQL define las siguientes tres operaciones de conjunto:

- **UNION**

Combina todos los resultados de dos bloques de consulta en un solo resultado, omitiendo duplicados.

```
SELECT columnas FROM tabla1 UNION SELECT columnas FROM tabla2;
# o
TABLE tabla1 UNION TABLE tabla2;
```

- **INTERSECT**

Combina solo las filas que los resultados de dos bloques de consulta tienen en común, omitiendo duplicados.

```
SELECT columnas FROM tabla1 INTERSECT SELECT columnas FROM tabla2;
# o
TABLE tabla1 INTERSECT TABLE tabla2;
```

- **EXCEPT**

Para dos bloques de consulta A y B, devuelve todos los resultados de A que no están presentes en B, omitiendo duplicados.

```
SELECT columnas FROM tabla1 EXCEPT SELECT columnas FROM tabla2;
# o
TABLE tabla1 EXCEPT TABLE tabla2;
```

Ejemplos:

Para terminar de entender como funciona las operaciones de conjunto vamos a ver el siguiente ejemplo:

```
DROP TEMPORARY TABLE IF EXISTS tabla_temporal1;
CREATE TEMPORARY TABLE tabla_temporal1 AS
VALUES ROW(10, 'Alice'), ROW(20, 'Bob'), ROW(30, 'Charlie');
DROP TEMPORARY TABLE IF EXISTS tabla_temporal2;
CREATE TEMPORARY TABLE tabla_temporal2 AS
VALUES ROW(20, 'Bob'), ROW(15, 'David'), ROW(25, 'Eve');
```

Si ejecutamos la unión de estas dos tablas el resultado serán todas sus filas sin repetir, en este caso ROW(20, 'Bob') solo aparecerá una vez

```
TABLE tabla_temporal1 UNION TABLE tabla_temporal2;
column_0    column_1
10          Alice
20          Bob
30          Charlie
15          David
25          Eve
```

Si ejecutamos la intersección el resultado será solo aquellas filas que aparecen en ambas tablas:

```
TABLE tabla_temporal1 INTERSECT TABLE tabla_temporal2;
column_0    column_1
20          Bob
```

Si ejecutamos la excepción el resultado será solo aquellas filas que aparecen en la primera tabla y no aparecen en la segunda tabla:

```
TABLE tabla_temporal1 EXCEPT TABLE tabla_temporal2;
column_0    column_1
10          Alice
30          Charlie

TABLE tabla_temporal2 EXCEPT TABLE tabla_temporal1;
column_0    column_1
15          David
25          Eve
```

Diferencias entre JOINs y las Operaciones de conjunto

- ***Operaciones de JOIN*:**
  - Combina filas de múltiples tablas basadas en una condición de igualdad.
  - Puede combinar tablas con estructuras de columna diferentes.
  - Puede generar filas duplicadas en el resultado.
  - Igualdad basada en una condición específica, no requiere igualdad en todas las columnas.
- ***Operaciones de conjunto (UNION, EXCEPT, INTERSECTION)*:**
  - Operan sobre resultados de consultas.
  - Requiere una estructura de columna idéntica en las tablas involucradas.
  - Elimina automáticamente filas duplicadas en el resultado.

- Igualdad basada en filas completas, no permite la especificación de columnas específicas para la igualdad.