TRABAJO TEORICO 3

Asignatura: Laboratorio IV
Trabajo: DDL *Data definition language* y DML *Data manipulation language*
Año: 2024
Alumno: Gallac Leandro Gabriel

1. Escriba el formato general de la sentencia `ALTER TABLE` y detalle las acciones

➔ El formato básico de la sentencia `ALTER TABLE` es:

```
ALTER TABLE nombre_tabla ACCION;
```

Algunas acciones comunes que se pueden realizar con `ALTER TABLE` son:

- **Agregar una columna**:
  `ALTER TABLE nombre_tabla ADD nombre_columna tipo_de_dato;`
- **Eliminar una columna**:
  `ALTER TABLE nombre_tabla DROP COLUMN nombre_columna;`
- **Modificar una columna** (por ejemplo, cambiar el tipo de dato o tamaño):
  `ALTER TABLE nombre_tabla MODIFY nombre_columna nuevo_tipo_de_dato;`
- **Renombrar una columna**:
  `ALTER TABLE nombre_tabla RENAME COLUMN nombre_columna_actual TO nuevo_nombre_columna;`
- **Agregar una clave foránea**:
  `ALTER TABLE nombre_tabla ADD CONSTRAINT nombre_constraint FOREIGN KEY (columna) REFERENCES otra_tabla(otra_columna);`

2. ¿Qué debo realizar antes de ejecutar un `ALTER TABLE` en una base de datos en producción?

➔ Antes de ejecutar un `ALTER TABLE` en una base de datos en producción, es importante:

- **Realizar un respaldo completo** de la base de datos para prevenir pérdidas en caso de errores.
- **Verificar el impacto** que el cambio puede tener en el sistema (consultas existentes, rendimiento, etc.).
- **Probar los cambios en un entorno de pruebas** antes de aplicarlos en producción.
- **Planificar el tiempo de ejecución**, ya que algunos cambios (por ejemplo, agregar o modificar columnas en tablas grandes) pueden tomar tiempo y afectar la disponibilidad del sistema.

3. Sintaxis de la sentencia para realizar consultas multitablas

➔ Para realizar consultas que involucren varias tablas (JOINs), la sintaxis general es:

```
SELECT columnas
FROM tabla1
[INNER | LEFT | RIGHT | FULL] JOIN tabla2
ON tabla1.columna_comun = tabla2.columna_comun
WHERE condiciones;
```

- Ejemplo de una consulta `JOIN` entre dos tablas:

```
SELECT empleados.nombre, departamentos.nombre
FROM empleados
INNER JOIN departamentos ON empleados.departamento_id =
departamentos.id;
```

4. ¿En qué caso utilizaría `UNIÓN`?

➔ Utilizaría la `UNION` cuando quiera combinar los resultados de dos o más consultas SQL distintas, siempre y cuando las consultas devuelvan el mismo número de columnas y tipos de datos compatibles.

- Ejemplo de uso:

```
SELECT nombre FROM clientes
UNION
SELECT nombre FROM proveedores;
```

Esto devolverá una lista única de nombres de clientes y proveedores, eliminando duplicados.

5. Sentencia que permita borrar un registro de la tabla Provincia aun cuando tiene referencias en la tabla Padrón.

➔ Para permitir borrar un registro en `Provincia` a pesar de las referencias en `Padrón`, podemos definir la clave foránea con la opción `ON DELETE CASCADE`. Esto implicaría

que al borrar un registro en `Provincia`, los registros relacionados en `Padrón` también se borrarán.

```
ALTER TABLE Padron
ADD CONSTRAINT fk_provincia_padron
FOREIGN KEY (provincia_id) REFERENCES Provincia(id)
ON DELETE CASCADE;
```

6. ¿En qué casos es necesario crear un índice?

➔ Los índices son útiles para mejorar el rendimiento de las consultas que involucran:

- **Búsquedas frecuentes** por columnas específicas.
- **Ordenamiento** (`ORDER BY`) y **agrupamiento** (`GROUP BY`).
- **Uniones** (`JOIN`) entre tablas donde se utilizan columnas específicas para relacionar.
- **Condiciones de filtrado** frecuentes con `WHERE`.

En lo posible hay que evitar crear índices en tablas con muchas operaciones de escritura (insertar, actualizar, eliminar), ya que los índices deben mantenerse actualizados.

7. ¿Qué sentencias sacan mayor provecho de los índices?

➔ Las sentencias que más se benefician de los índices son:

- **`SELECT` con filtros** (`WHERE`) que usan las columnas indexadas.
- **Consultas con `JOIN`** sobre columnas indexadas.
- **Consultas que usan `ORDER BY` o `GROUP BY`** en columnas indexadas.

8. Pros y contras de crear índices

➔ **Pros**:

- ◆ Mejora el rendimiento de las consultas de búsqueda.
- ◆ Acelera las operaciones de `JOIN`, `ORDER BY`, y `GROUP BY`.

➔ **Contras**:

- ◆ Ocupa espacio en disco adicional.
- ◆ Ralentiza las operaciones de escritura (inserciones, actualizaciones, eliminaciones) debido a la necesidad de actualizar los índices.

9. ¿Cómo se puede saber qué índices se están utilizando para resolver una consulta?

➔ Podemos utilizar el comando `EXPLAIN` o `EXPLAIN ANALYZE` para ver cómo el motor de la base de datos resuelve una consulta y qué índices está usando.

```
EXPLAIN SELECT * FROM clientes WHERE id = 1;
```

Esto mostrará el plan de ejecución de la consulta y si se está usando algún índice.

10. Sentencia para ver los nombres de los índices de una tabla

➔ Para obtener los nombres de los índices de una tabla, puedes usar:

```
SHOW INDEX FROM nombre_tabla;
```

Esto devolverá una lista de los índices asociados con esa tabla, incluyendo el nombre y la columna a la que se aplica.