Escrito por **Adrián Quiroga Linares**.

7.1 Transacciones

En bases de datos, una **transacción** es una secuencia de operaciones SQL que se ejecutan como una unidad de trabajo. Las transacciones garantizan que las operaciones sean **atómicas** (o se completan todas, o ninguna) y preservan la **consistencia** de la base de datos.

1. **Inicio de la Transacción**: En SQL, una transacción se inicia automáticamente al ejecutar una sentencia SQL. No es necesario un comando específico para iniciar una transacción; cualquier consulta de actualización (como `UPDATE`, `INSERT`, o `DELETE`) marca implícitamente el comienzo.
2. **Finalización de la Transacción**: Para finalizar una transacción, SQL ofrece dos comandos principales:
   - `COMMIT`: Guarda todos los cambios realizados durante la transacción y los hace permanentes en la base de datos. Se garantiza que todos los cambios sean visibles para otros usuarios.
   - `ROLLBACK`: Revierte todos los cambios de la transacción, dejando la base de datos en el estado en el que se encontraba antes de iniciar la transacción. Esto es útil para deshacer operaciones en caso de errores.

   Por ejemplo, si realizas varias operaciones en una aplicación bancaria para transferir fondos entre dos cuentas:

   ```sql
   BEGIN; -- (Iniciado implícitamente al realizar la primera operación)
   UPDATE cuentas SET saldo = saldo - 100 WHERE cuenta_id = 1; -- Restar dinero
   UPDATE cuentas SET saldo = saldo + 100 WHERE cuenta_id = 2; -- Sumar dinero
   COMMIT; -- Guardar los cambios
   ```

   Si hay algún error entre las actualizaciones (por ejemplo, si se retira dinero pero falla al sumarlo), el `ROLLBACK` garantiza que la base de datos regrese al estado inicial, evitando inconsistencias.
3. **Fallos y Recuperación Automática**: Si ocurre un fallo (por ejemplo, una caída de sistema) antes de ejecutar `COMMIT`, SQL garantiza que los efectos de la transacción incompleta se reviertan automáticamente. Al reiniciar, el sistema restaura la base de datos al estado anterior de la transacción fallida.

7.2 Extensiones a la creación de tablas

Crear una tabla a partir de otra existente La sentencia `CREATE TABLE ... LIKE` se utiliza para crear una nueva tabla que tiene el mismo esquema (estructura de columnas y tipos de datos) que una tabla existente. Por ejemplo:

```sql
CREATE TABLE temp_profesor LIKE profesor;
```

En este caso, `temp_profesor` se crea con la misma estructura que la tabla `profesor`, pero sin ningún dato.

Crear una tabla a partir de una consulta

La sentencia `CREATE TABLE ... AS` permite crear una nueva tabla y llenarla con los resultados de una consulta. Por ejemplo:

```
CREATE TABLE t1 AS
SELECT * FROM profesor WHERE nombre_dept = 'Música' WITH DATA;
```

En este caso, `t1` se crea y se llena con todas las filas de la tabla `profesor` donde la columna `nombre_dept` tiene el valor 'Música'. La cláusula `WITH DATA` indica que la tabla se rellena con los resultados de la consulta. Si omites `WITH DATA`, según la norma SQL:2003, la tabla se crea vacía:

```
CREATE TABLE t2 AS
SELECT * FROM profesor WHERE nombre_dept = 'Música';
```

7.3 Autorización de privilegios

La autorización en bases de datos implica asignar diferentes tipos de permisos a los usuarios sobre las partes de la base de datos. Estos permisos se conocen como **privilegios** y se clasifican principalmente en cuatro tipos:

1. **Autorización de lectura de datos** (SELECT)
2. **Autorización de inserción de nuevos datos** (INSERT)
3. **Autorización de actualización de datos** (UPDATE)
4. **Autorización de borrado de datos** (DELETE)

- **Lectura**: Un usuario llamado `profesor1` puede consultar datos de la tabla `estudiantes`: `sql GRANT SELECT ON estudiantes TO profesor1;`
- **Inserción**: Un usuario llamado `secretaria` puede agregar nuevos registros a la tabla: `sql GRANT INSERT ON estudiantes TO secretaria;`
- **Actualización**: Un usuario llamado `profesor2` puede actualizar el correo electrónico de los estudiantes: `sql GRANT UPDATE (email) ON estudiantes TO profesor2;`
- **Borrado**: Un usuario llamado `admin` puede eliminar registros de la tabla: `sql GRANT DELETE ON estudiantes TO admin;`

7.4. Revocación de Privilegios

Los privilegios se conceden utilizando la sentencia `GRANT` y se revocan utilizando la sentencia `REVOKE`. Por ejemplo:

- Para conceder múltiples privilegios a un usuario: `sql GRANT SELECT, INSERT, UPDATE ON estudiantes TO profesor1;`
- Para revocar un privilegio: `sql REVOKE UPDATE ON estudiantes FROM profesor2;`

7.5 Roles

Los **roles** son grupos de privilegios que se pueden asignar a usuarios. Esto es útil cuando varios usuarios requieren los mismos permisos. Por ejemplo, en una universidad, podríamos tener roles como `profesor`, `estudiante`, o `administrador`.

- **Creación de un rol**: `sql CREATE ROLE profesor;`
- **Concesión de privilegios al rol**: `sql GRANT SELECT ON estudiantes TO profesor;`
- **Asignación de roles a usuarios**: `sql GRANT profesor TO profesor1;`

Cuando un usuario se conecta a la base de datos, tiene todos los privilegios que se le han concedido directamente, así como los que hereda de cualquier rol que se le haya asignado.