SQL Cheatsheet

Grupo 11

19 de junio de 2024

Índice

1. Introducción

Este documento contiene un resumen de las principales sentencias SQL necesarias para trabajar con bases de datos en PostgreSQL. Se incluyen ejemplos prácticos y explicaciones detalladas para cada tipo de operación.

2. psql

`psql` es la interfaz de línea de comandos de PostgreSQL. Se utiliza para interactuar directamente con la base de datos desde la terminal.

2.1. Conectar a la Base de Datos

Para conectarse a una base de datos, utiliza el comando:

```
psql -h hostname -U username -d dbname
```

Tips: - Reemplaza `hostname` con la dirección del servidor, `username` con tu nombre de usuario, y `dbname` con el nombre de la base de datos. - Usa `psql -h localhost -U postgres -d mi_base_de_datos` para conectarte a una base de datos local.

2.2. Ejecutar Comandos SQL desde psql

Puedes ejecutar cualquier comando SQL directamente desde `psql`:

```
psql -c "SELECT␣*␣FROM␣tabla;"
```

Tips: - Usa comillas dobles para encapsular el comando SQL.

2.3. Salir de psql

Para salir de la interfaz `psql`, usa:

```
\q
```

3. Crear Tablas

3.1. Sintaxis Básica

Esto se utiliza para definir una nueva tabla en la base de datos con los campos especificados. Se recomienda utilizar para definir la estructura inicial de los datos que desea almacenar.

```
CREATE TABLE Peliculas (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    ano INT,
    categoria VARCHAR(50),
    calificacion DECIMAL(3,1),
    director VARCHAR(100)
);
```

Tips: - Asegúrate de definir la clave primaria para identificar de manera única cada fila. - Utiliza tipos de datos apropiados para cada columna.

3.2. Tabla Temporal

Esto se utiliza para crear una tabla que solo existe durante la sesión actual. Se recomienda utilizar para almacenar datos temporales o resultados intermedios.

```
CREATE TEMP TABLE TempPeliculas (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    ano INT,
    categoria VARCHAR(50),
    calificacion DECIMAL(3,1),
    director VARCHAR(100)
);
```

Tips: - Útil para operaciones de procesamiento de datos que no necesitan ser persistentes. - Las tablas temporales se eliminan automáticamente al final de la sesión.

3.3. Tabla con Llaves Foráneas

Esto se utiliza para definir relaciones entre diferentes tablas mediante llaves foráneas. Se recomienda utilizar para mantener la integridad referencial en la base de datos.

```
CREATE TABLE Actores (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    edad INT
);

CREATE TABLE Actuo_en (
    id_actor INT,
    id_pelicula INT,
    FOREIGN KEY (id_actor) REFERENCES Actores(id),
    FOREIGN KEY (id_pelicula) REFERENCES Peliculas(id),
    PRIMARY KEY (id_actor, id_pelicula)
);
```

Tips: - Define llaves foráneas para asegurar que los datos relacionados existan en las tablas correspondientes. - Utiliza la cláusula ‘ON DELETE CASCADE‘ si deseas que la eliminación de una fila en la tabla principal elimine las filas correspondientes en la tabla relacionada.

4. Insertar Datos

4.1. Insertar en una Tabla

Esto se utiliza para agregar una nueva fila a la tabla especificada. Se recomienda utilizar para añadir nuevos registros a una tabla existente.

```
INSERT INTO Peliculas (nombre, ano, categoria, calificacion, director)
VALUES ('Inception', 2010, 'Adventure', 8.8, 'C. Nolan');
```

Tips: - Siempre especifica las columnas en las que deseas insertar datos para evitar errores si la estructura de la tabla cambia. - Asegúrate de que los valores proporcionados coincidan con los tipos de datos de las columnas.

4.2. Insertar Múltiples Filas

Esto se utiliza para insertar varias filas en una tabla en una sola sentencia. Se recomienda utilizar para mejorar el rendimiento al insertar grandes volúmenes de datos.

```
INSERT INTO Peliculas (nombre, ano, categoria, calificacion, director)
VALUES
('Avatar', 2009, 'Sci-Fi', 7.8, 'J. Cameron'),
('The Dark Knight', 2008, 'Action', 9.0, 'C. Nolan');
```

Tips: - Inserta múltiples filas a la vez para reducir el número de transacciones y mejorar la eficiencia.

4.3. Insertar en una Tabla Temporal

Esto se utiliza para agregar datos a una tabla temporal. Se recomienda utilizar para almacenar resultados intermedios o datos temporales durante el procesamiento de consultas.

```
INSERT INTO TempPeliculas (nombre, ano, categoria, calificacion, director)
VALUES ('Avatar', 2009, 'Sci-Fi', 7.8, 'J. Cameron');
```

Tips: - Útil en procedimientos almacenados y scripts de análisis de datos.

4.4. Insertar con Llaves Foráneas

Esto se utiliza para agregar datos a una tabla que tiene relaciones de llaves foráneas. Se recomienda utilizar para asegurar que los datos relacionados existan en las tablas correspondientes.

```
INSERT INTO Actores (nombre, edad) VALUES ('Leonardo DiCaprio', 46);
INSERT INTO Actuo_en (id_actor, id_pelicula) VALUES (1, 1);
```

Tips: - Verifica que las llaves foráneas existan en las tablas principales antes de realizar la inserción.

5. Actualizar Datos

5.1. Actualizar una Tupla

Esto se utiliza para modificar los valores de una fila existente en una tabla. Se recomienda utilizar para corregir o actualizar la información en la base de

datos.

```
UPDATE Peliculas
SET calificacion = 9.0
WHERE nombre = 'Inception';
```

Tips: - Siempre incluye una cláusula ‘WHERE‘ para evitar actualizar todas las filas de la tabla inadvertidamente. - Utiliza ‘RETURNING‘ para ver los cambios realizados.

5.2. Actualizar Múltiples Filas

Esto se utiliza para actualizar varias filas en una tabla que cumplen con una condición específica. Se recomienda utilizar para realizar actualizaciones masivas basadas en ciertas condiciones.

```
UPDATE Peliculas
SET calificacion = calificacion + 0.1
WHERE ano >= 2010;
```

Tips: - Asegúrate de que la condición ‘WHERE‘ sea precisa para evitar actualizaciones no deseadas.

6. Eliminar Datos

6.1. Eliminar una Tupla

Esto se utiliza para eliminar una fila específica de una tabla. Se recomienda utilizar para remover registros obsoletos o incorrectos.

```
DELETE FROM Peliculas
WHERE nombre = 'Avatar';
```

Tips: - Incluye siempre una cláusula ‘WHERE‘ para evitar eliminar todas las filas de la tabla.

6.2. Eliminar con Condiciones

Esto se utiliza para eliminar varias filas que cumplen con una condición específica. Se recomienda utilizar para limpiar datos que cumplen ciertos criterios.

```
DELETE FROM Peliculas
WHERE calificacion < 7.0;
```

Tips: - Verifica las condiciones de eliminación para asegurarte de que solo se eliminen las filas deseadas.

7. Crear un Trigger

Esto se utiliza para definir acciones automáticas que se ejecutan cuando ocurre un evento específico en la base de datos. Se recomienda utilizar para mantener la consistencia de los datos y automatizar procesos.

```
CREATE OR REPLACE FUNCTION actualizar_calificacion() RETURNS TRIGGER AS $$
BEGIN
    NEW.calificacion := ROUND(NEW.calificacion, 1);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_actualizar_calificacion
BEFORE INSERT OR UPDATE ON Peliculas
FOR EACH ROW EXECUTE FUNCTION actualizar_calificacion();
```

Tips: - Utiliza triggers para validar datos y mantener la integridad referencial. - Asegúrate de que los triggers no introduzcan ciclos infinitos o bloqueos en la base de datos.

8. Crear un Procedimiento Almacenado

8.1. Insertar Datos con un Procedimiento Almacenado

Esto se utiliza para definir un conjunto de instrucciones SQL que se almacenan en la base de datos y se pueden ejecutar bajo demanda. Se recomienda utilizar para encapsular lógica compleja y reutilizable.

```
CREATE OR REPLACE PROCEDURE insertar_pelicula(
    nombre VARCHAR,
    ano INT,
    categoria VARCHAR,
    calificacion DECIMAL,
    director VARCHAR
) LANGUAGE plpgsql AS $$
BEGIN
    INSERT INTO Peliculas (nombre, ano, categoria, calificacion, director)
    VALUES (nombre, ano, categoria, calificacion, director);
END;
$$;

CALL insertar_pelicula('Interstellar', 2014, 'Sci-Fi', 8.6, 'C. Nolan');
```

Tips: - Utiliza procedimientos almacenados para operaciones repetitivas y complejas. - Mejora la seguridad y el rendimiento de las consultas.

8.2. Actualizar Datos con un Procedimiento Almacenado

Esto se utiliza para actualizar datos en la base de datos mediante un procedimiento almacenado. Se recomienda utilizar para actualizar múltiples registros de manera eficiente.

```
CREATE OR REPLACE PROCEDURE actualizar_calificacion_pelicula(
    pelicula_id INT,
    nueva_calificacion DECIMAL
) LANGUAGE plpgsql AS $$
BEGIN
    UPDATE Peliculas
    SET calificacion = nueva_calificacion
    WHERE id = pelicula_id;
END;
$$;

CALL actualizar_calificacion_pelicula(1, 9.5);
```

Tips: - Encapsula la lógica de actualización en procedimientos almacenados para facilitar el mantenimiento.

9. Crear una Vista

9.1. Vista Simple

Esto se utiliza para crear una vista, que es una consulta almacenada que puede ser tratada como una tabla. Se recomienda utilizar para simplificar consultas complejas y reutilizarlas.

```
CREATE VIEW VistaPeliculas AS
SELECT nombre, director, calificacion
FROM Peliculas
WHERE calificacion > 8.0;
```

Tips: - Utiliza vistas para encapsular consultas frecuentes y reducir la complejidad. - Las vistas no almacenan datos, solo la consulta.

9.2. Vista con Join

Esto se utiliza para crear una vista que combine datos de múltiples tablas. Se recomienda utilizar para presentar datos relacionados en una sola estructura.

```
CREATE VIEW VistaPeliculasActores AS
SELECT Peliculas.nombre AS pelicula, Actores.nombre AS actor
FROM Peliculas
JOIN Actuo_en ON Peliculas.id = Actuo_en.id_pelicula
JOIN Actores ON Actuo_en.id_actor = Actores.id;
```

Tips: - Las vistas con joins son útiles para combinar y presentar datos de varias tablas de manera lógica.

9.3. Vista Materializada

Esto se utiliza para crear una vista que almacena físicamente los resultados de una consulta. Se recomienda utilizar para mejorar el rendimiento de consultas que se ejecutan frecuentemente.

```
CREATE MATERIALIZED VIEW VistaPeliculasMat AS
SELECT nombre, director, calificacion
FROM Peliculas
WHERE calificacion > 8.0;

-- Para actualizar la vista materializada
REFRESH MATERIALIZED VIEW VistaPeliculasMat;
```

Tips: - Las vistas materializadas son útiles para grandes conjuntos de datos que no cambian con frecuencia. - Recuerda refrescar la vista para mantenerla actualizada.

10. Consultas Básicas

10.1. Seleccionar Todos los Registros

Esto se utiliza para recuperar todas las filas de una tabla. Se recomienda utilizar para obtener una vista completa de los datos en una tabla.

```
SELECT * FROM Peliculas;
```

Tips: - Útil para inspeccionar el contenido completo de una tabla. - Evita usar en tablas muy grandes sin filtros adecuados.

10.2. Seleccionar con Condiciones

Esto se utiliza para recuperar filas que cumplen con una condición específica. Se recomienda utilizar para filtrar datos según criterios específicos.

```
SELECT nombre, calificacion
FROM Peliculas
WHERE director = 'C. Nolan';
```

Tips: - Utiliza operadores de comparación ($>$, $<$, $=$, etc.) para definir condiciones precisas. - Combina condiciones con AND y OR para mayor flexibilidad.

10.3. Seleccionar con LIKE

Esto se utiliza para buscar filas donde una columna coincide parcialmente con un patrón. Se recomienda utilizar para realizar búsquedas flexibles de texto.

```
SELECT * FROM Peliculas
WHERE nombre ILIKE '%terminator%';
```

Tips: - Utiliza % como comodín para cualquier secuencia de caracteres. - Utiliza _ como comodín para un solo carácter.

10.4. Seleccionar con IN

Esto se utiliza para seleccionar filas donde el valor de una columna coincide con cualquiera de una lista de valores. Se recomienda utilizar para consultas donde se conocen varios valores posibles.

```
SELECT * FROM Peliculas
WHERE ano IN (2010, 2014);
```

Tips: - IN es útil para comparar una columna con un conjunto de valores específicos.

11. Uniones y Joins

11.1. Inner Join

Esto se utiliza para combinar filas de dos o más tablas basadas en una condición de coincidencia. Se recomienda utilizar para combinar datos relacionados de múltiples tablas.

```
SELECT Peliculas.nombre, Actores.nombre
FROM Peliculas
JOIN Actuo_en ON Peliculas.id = Actuo_en.id_pelicula
JOIN Actores ON Actuo_en.id_actor = Actores.id;
```

Tips: - Inner Join devuelve solo las filas que tienen coincidencias en ambas tablas. - Útil para relaciones muchos a muchos.

11.2. Left Join

Esto se utiliza para combinar filas de dos o más tablas, devolviendo todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha. Se recomienda utilizar para obtener todas las filas de una tabla, incluso si no hay coincidencias en la otra tabla.

```
SELECT Peliculas.nombre, Actores.nombre
FROM Peliculas
LEFT JOIN Actuo_en ON Peliculas.id = Actuo_en.id_pelicula
LEFT JOIN Actores ON Actuo_en.id_actor = Actores.id;
```

Tips: - Left Join es útil para obtener todas las filas de la tabla principal y los datos relacionados cuando existan.

11.3. Right Join

Esto se utiliza para combinar filas de dos o más tablas, devolviendo todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda. Se recomienda utilizar para obtener todas las filas de una tabla secundaria, incluso si no hay coincidencias en la otra tabla.

```
SELECT Peliculas.nombre, Actores.nombre
FROM Peliculas
RIGHT JOIN Actuo_en ON Peliculas.id = Actuo_en.id_pelicula
RIGHT JOIN Actores ON Actuo_en.id_actor = Actores.id;
```

Tips: - Right Join es útil cuando necesitas todas las filas de la tabla secundaria.

11.4. Full Outer Join

Esto se utiliza para combinar filas de dos o más tablas, devolviendo todas las filas cuando hay una coincidencia en una de las tablas. Se recomienda utilizar para obtener todas las filas cuando hay coincidencias en cualquiera de las tablas.

```
SELECT Peliculas.nombre, Actores.nombre
FROM Peliculas
FULL OUTER JOIN Actuo_en ON Peliculas.id = Actuo_en.id_pelicula
FULL OUTER JOIN Actores ON Actuo_en.id_actor = Actores.id;
```

Tips: - Full Outer Join devuelve todas las filas cuando hay coincidencias en una de las tablas. - Útil para análisis completos donde se necesita toda la información.

12. Agregación

12.1. Funciones de Agregación

Esto se utiliza para realizar cálculos en un conjunto de valores y devolver un solo valor. Se recomienda utilizar para obtener estadísticas resumidas de los datos.

```
SELECT AVG(calificacion) AS promedio_calificacion
FROM Peliculas;
```

Tips: - Utiliza funciones como AVG, SUM, MIN, MAX, COUNT para agregar datos. - Combinable con GROUP BY para agregaciones por grupo.

12.2. Agrupar por y Agregar

Esto se utiliza para agrupar filas que tienen los mismos valores en columnas específicas y realizar cálculos agregados en cada grupo. Se recomienda utilizar para resumir datos en grupos.

```sql
SELECT director, COUNT(*) AS numero_peliculas
FROM Peliculas
GROUP BY director;
```

Tips: - GROUP BY se usa para crear un conjunto de resultados agrupados. - Útil para análisis de datos categorizados.

12.3. Having Clause

Esto se utiliza para filtrar grupos de resultados después de aplicar la cláusula GROUP BY. Se recomienda utilizar para aplicar condiciones a grupos agregados.

```sql
SELECT director, COUNT(*) AS numero_peliculas
FROM Peliculas
GROUP BY director
HAVING COUNT(*) > 1;
```

Tips: - HAVING se usa después de GROUP BY para filtrar los resultados agregados. - Similar a WHERE pero se aplica a los resultados agrupados.

13. Subconsultas

13.1. Subconsulta en SELECT

Esto se utiliza para incorporar una consulta dentro de otra consulta. Se recomienda utilizar para obtener valores calculados que dependen de otra consulta.

```sql
SELECT nombre, (SELECT AVG(calificacion) FROM Peliculas) AS promedio_general
FROM Peliculas;
```

Tips: - Las subconsultas en SELECT se utilizan para calcular valores en columnas.

13.2. Subconsulta en WHERE

Esto se utiliza para incorporar una consulta dentro de la cláusula WHERE. Se recomienda utilizar para filtrar resultados basados en otra consulta.

```sql
SELECT nombre
FROM Peliculas
WHERE calificacion > (SELECT AVG(calificacion) FROM Peliculas);
```

Tips: - Las subconsultas en WHERE se utilizan para comparar valores de la consulta principal con resultados de una subconsulta.

14. Otras Operaciones

14.1. Ordenar Resultados

Esto se utiliza para ordenar los resultados de una consulta en orden ascendente o descendente. Se recomienda utilizar para presentar datos en un orden específico.

```
SELECT nombre, calificacion
FROM Peliculas
ORDER BY calificacion DESC;
```

Tips: - Utiliza ORDER BY para ordenar los resultados según una o más columnas. - Especifica ASC para ascendente y DESC para descendente.

14.2. Limitar Resultados

Esto se utiliza para limitar el número de filas devueltas por una consulta. Se recomienda utilizar para obtener un subconjunto específico de resultados.

```
SELECT nombre, calificacion
FROM Peliculas
ORDER BY calificacion DESC
LIMIT 5;
```

Tips: - LIMIT es útil para paginación y para obtener los primeros N resultados.

14.3. Actualizar Datos en una Vista Materializada

Esto se utiliza para actualizar los datos almacenados en una vista materializada. Se recomienda utilizar para mantener actualizadas las vistas materializadas.

```
REFRESH MATERIALIZED VIEW VistaPeliculasMat;
```

Tips: - Actualiza las vistas materializadas regularmente para reflejar los datos más recientes.

15. Seguridad y Transacciones

15.1. Creación de Usuarios

Esto se utiliza para crear un nuevo usuario en la base de datos. Se recomienda utilizar para gestionar el acceso y la seguridad de la base de datos.

```
CREATE USER nuevo_usuario WITH PASSWORD 'contrasena';
```

Tips: - Define roles y permisos adecuados para los usuarios.

15.2. Asignación de Permisos

Esto se utiliza para otorgar permisos específicos a un usuario. Se recomienda utilizar para controlar el acceso a diferentes partes de la base de datos.

```
GRANT SELECT, INSERT, UPDATE ON TABLE Peliculas TO nuevo_usuario;
```

Tips: - Asigna permisos mínimos necesarios para cumplir con el principio de menor privilegio.

15.3. Uso de Transacciones

Esto se utiliza para agrupar varias operaciones en una transacción que se puede confirmar o revertir. Se recomienda utilizar para asegurar la integridad de los datos en operaciones complejas.

```
BEGIN;
INSERT INTO Peliculas (nombre, ano, categoria, calificacion, director)
VALUES ('Dunkirk', 2017, 'War', 7.9, 'C. Nolan');
UPDATE Peliculas
SET calificacion = 8.0
WHERE nombre = 'Dunkirk';
COMMIT;
```

Tips: - Utiliza transacciones para asegurar que un conjunto de operaciones se realice completamente o no se realice en absoluto. - Usa 'ROLLBACK' para deshacer una transacción en caso de error.