📚 Índices en Bases de Datos

📌 ¿Qué es un índice en bases de datos?

Un **índice** es una estructura de datos que mejora la velocidad de búsqueda en una tabla, similar a un índice en un libro. Permite acceder rápidamente a los registros sin escanear toda la tabla.

🚀 ¿Para qué sirven los índices?

1. **Aceleran consultas** (`SELECT` con `WHERE`, `ORDER BY`, `GROUP BY`).
2. **Mejoran el rendimiento en búsquedas**.
3. **Facilitan la búsqueda en columnas clave**.
4. **Reducen la carga en el sistema al evitar escaneos completos de tablas**.

◆ Tipos de Índices

1. **Índice Primario (`PRIMARY KEY`)**
   - Se crea automáticamente en la clave primaria.

```sql
CREATE TABLE Estudiantes (
    ID INT PRIMARY KEY,
    Nombre VARCHAR(50),
    Carrera VARCHAR(50)
);
```

2. **Índice Único (`UNIQUE`)**
   - Evita valores duplicados.

```sql
CREATE UNIQUE INDEX idx_email ON Usuarios(Email);
```

3. **Índice Compuesto**
   - Optimiza consultas con múltiples filtros.

🔥 Ejemplo de Creación y Uso de Índices

Paso 1: Crear la Tabla y Agregar Datos

```
CREATE TABLE Clientes (
    ID SERIAL PRIMARY KEY,
    Nombre VARCHAR(50),
    Apellido VARCHAR(50),
    Email VARCHAR(100) UNIQUE
);

INSERT INTO Clientes (Nombre, Apellido, Email) VALUES
('Juan', 'González', 'juan@example.com'),
('Ana', 'Pérez', 'ana@example.com'),
('Luis', 'Fernández', 'luis@example.com');
```

Paso 2: Crear un Índice en la Columna `Apellido`

```
CREATE INDEX idx_apellido ON Clientes(Apellido);
```

💡 **Beneficio**: Este índice optimiza las búsquedas por apellido.

Paso 3: Verificar el Uso del Índice con `EXPLAIN ANALYZE`

```
EXPLAIN ANALYZE SELECT * FROM Clientes WHERE Apellido = 'González';
```

Salida esperada si usa el índice:

```
Index Scan using idx_apellido on clientes  (cost=0.15..4.30 rows=1 width=50)
```

- `Index Scan` indica que el índice está en uso, mejorando la consulta.
- **Si aparece** `Seq Scan`, significa que no se está usando el índice, lo que puede indicar problemas de configuración.

◆ Buenas Prácticas al Usar Índices

- ✓ Usar índices en columnas que se buscan frecuentemente (`WHERE`, `JOIN`).
- ✓ Evitar índices en columnas con muchos valores repetidos.
- ✓ No crear demasiados índices, ya que afectan `INSERT`, `UPDATE`, `DELETE`.
- ✓ Usar índices compuestos solo cuando haya consultas con múltiples filtros.
- ✓ Actualizar estadísticas de índices regularmente para mantener eficiencia.

🚀 Conclusión

- ✔ Los índices mejoran la velocidad de búsqueda y consultas.
- ✔ PostgreSQL permite analizar su uso con `EXPLAIN ANALYZE`.
- ✔ Es importante diseñar los índices estratégicamente para optimizar el rendimiento.