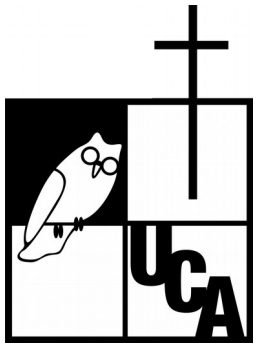
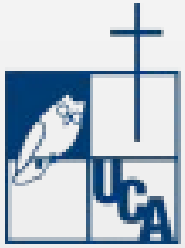


Structured Query Language

Data Manipulation Language

Administración de Bases de Datos



Departamento de
electrónica e informática

Carlos Juan Martín Pérez

EXPLAIN: Comando SQL que describe la ejecución de una sentencia SQL.

EXPLAIN ANALYZE: Agrega a la descripción información sobre el tiempo de planificación de la consulta y el de ejecución.

- **Seq Scan:** menor esfuerzo de planificación, mayor esfuerzo de ejecución, recorrido total de la tabla en busca de resultados para ser devueltos/procesados.
- **Index Scan:** mayor esfuerzo de planificación, menor esfuerzo de ejecución, recorrido a través de estructura de índice en busca de resultados para ser devueltos/procesados.

```
explain analyze select * from proyecto where
  codigo='00005';

explain analyze select * from proyecto where
  denominacion='aulavirt';

¡Más tiempo usando un índice!... ¿paradoja?: no tanto,
  depende del tamaño de la tabla de datos a consultar.
```

CREATE INDEX: Comando SQL que describe la ejecución de una sentencia SQL. Notas: cuando se crea un Primary Key automáticamente se genera un índice. Se usa DROP INDEX para eliminarlos.

```
explain analyze select * from muchosclientes where
  telefono='223664661973';
explain analyze select * from muchosclientes where id='81';
```

```
CREATE INDEX idx_muchosclientes_telefono
ON muchosclientes(telefono);
```

```
explain analyze select * from muchosclientes where
  telefono='223664661973';
```

```
¡Las búsquedas son más eficientes! pero nada es gratis en la
  vida: cada insert, update o delete provocará tiempo de
  recálculo del índice, luego tendrá que valorarse el costo,
  sobre todo si la tabla se actualiza con frecuencia.
```

Tipos de índices: B-tree, Hash, GiST, SP-GiST, GIN y BRIN. Cada tipo de índice utiliza una estructura de almacenamiento y un algoritmo de búsqueda diferente para efectuar diferentes tipos (especializados) de consultas. Si no se especifica, se crea un B-tree. https://www.postgresql.org/docs/current/indexes-types.html

B-tree: árbol de búsqueda balanceado. Se usa cuando la consulta utiliza operadores de comparación (<,>, <=, =, >=), BETWEEN, IN, IS NULL, IS NOT NULL y también LIKE si el patrón de búsqueda es constante en el inicio (p.e. 'Periqui%')

Hash: solo se usa cuando la consulta se expresa con operador de igualdad (=), por lo que no son muchos casos. En estos casos funciona mejor que el B-tree si las columnas tienen restricción UNIQUE.

GIN: índices generales invertidos. Son prácticos cuando en la columna se almacenan múltiples valores (rangos, arrays, jsonb)

BRIN: se usan en tablas de tamaño inmenso (B-tree sería muy costoso), generalmente en columnas que están ordenadas linealmente (p.e. fechas)

GiST: son útiles para indexar datos geoespaciales y búsquedas en full-text.

SP-GiST: se utilizan para hacer más eficientes búsquedas en datos ya agrupados naturalmente en estructuras no balanceadas: p.e. tablas de enrutamiento.

Índices sobre expresiones: generalmente los índices se establecen sobre una columna, pero también se pueden utilizar expresiones sobre la columna. Veamos un ejemplo MUY común:

```
SELECT nombre,apellidos,email,telefono FROM muchosclientes
    WHERE nombre='Andrea';
-- Hagamos EXPLAIN ANALYZE de la anterior consulta
CREATE INDEX idx_muchosclientes_nombre ON
    muchosclientes(nombre);
-- Veamos cuanto tarda ahora ¡bien! y... ¿será igual con la siguiente?
SELECT nombre,apellidos,email,telefono FROM muchosclientes
    WHERE lower(nombre)=lower('Andrea');
-- Es un horror! Y no se usa el índice
DROP INDEX idx_muchosclientes_nombre;
CREATE INDEX idx_muchosclientes_nombre ON muchosclientes
    (lower(nombre));
-- ¡Ahora sí!
```

Índices multicolumna: los índices se suelen establecer sobre una columna, pero también se pueden utilizar varias, y es MUY importante considerar el orden de ellas en la definición del índice. Veamos otro ejemplo:

```
SELECT nombre,apellidos,email,telefono FROM muchosclientes
    WHERE lower(nombre)=lower('Andrea') and
    lower(apellidos)=lower('Henderson');
-- Hagamos EXPLAIN ANALYZE de la anterior consulta
CREATE INDEX idx_muchosclientes_nombreapellidos ON
    muchosclientes (lower(nombre),lower(apellidos));
-- Veamos cuanto tarda ahora ¡bien! y... ¿será igual con la siguiente?
SELECT nombre,apellidos,email,telefono FROM muchosclientes
    WHERE lower(nombre)=lower('Andrea');
-- ¿Y con ésta?
SELECT nombre,apellidos,email,telefono FROM muchosclientes
    WHERE lower(apellidos)=lower('Henderson');
-- Ojo: el índice se usa solo según el orden de las columnas
```

Índices parciales: en algunas ocasiones podemos hacer más eficientes las consultas si consideramos el descarte de datos no muy consultados (utilizando índices de un tamaño más reducido). Probemos:

```
EXPLAIN ANALYZE SELECT id,nombre,apellidos,email,telefono FROM muchosclientes WHERE activo = false;
CREATE INDEX idx_muchosclientes_actividad ON muchosclientes (activo);
-- podemos hacerlo más eficiente excluyendo a los clientes activos (la mayoría)
CREATE INDEX idx_muchosclientes_actividad ON muchosclientes (activo) WHERE activo = false;
-- Ojo ¡puede que no se esté usando el índice!: este tipo de índice es útil en tablas grandes (el planificador de consultas evalúa si lo usa o no). Veamos con este otro caso:
create index idx_muchosusuarios_standard on muchosusuarios(id) where user_type = 'Standard';
explain analyze select * from muchosusuarios where user_type = 'Standard';
```

REINDEX: en ocasiones (raras) es necesario reconstruir algún índice (o todos). NO es lo mismo que hacer DROP y CREATE INDEX: REINDEX solo bloquea escrituras mientras se construye, DROP INDEX bloquea lecturas y escrituras y CREATE INDEX las escrituras (pero las lecturas pueden ser lentas mientras se crea el índice).

```
REINDEX INDEX idx_muchosclientes_actividad;

REINDEX (VERBOSE) DATABASE ucasoft;
```

Transacciones: una transacción es un conjunto de sentencias que deben ejecutarse respetando el criterio ACID (atomicidad, consistencia, aislamiento, durabilidad). Ejemplo típico: transferencia de dinero entre cuentas.

```
BEGIN; (o BEGIN WORK, BEGIN TRANSACTION)
-- hacer consultas….
COMMIT; -- si todo está en orden
ROLLBACK; -- si hay que dar marcha atrás
```

También puede ser necesario agrupar sentencias en una transacción (y establecer algunas condiciones particulares a la aplicación de reglas de integridad dentro de la transacción) por situaciones especiales, por ejemplo al insertar datos en tablas con referencias circulares → Ver caso del labo 2.