
Bases de Datos

SQL

Características generales

Lenguaje estructurado

```
1 CREATE TABLE paises(  
2   codigo CHAR(2) PRIMARY KEY, —————> Declaración de clave primaria  
3   nombre TEXT UNIQUE); —————> No se aceptan valores repetidos  
4  
5 CREATE TABLE ciudades(  
6   nombre TEXT,  
7   codigo_postal VARCHAR(9) CHECK (codigo_postal <> ''), —————> No se aceptan NULL  
8   PRIMARY KEY (nombre, codigo_postal)); —————> Declaración de clave  
9                                           primaria compuesta  
10 CREATE TABLE delivery(  
11   ID_delivery INTEGER PRIMARY KEY,  
12   codigo_inicio VARCHAR(9),  
13   codigo_llegada VARCHAR(9));
```

CRUD

Create, Read, Update, Delete

```
1  INSERT INTO paises VALUES ('US', 'United States'),('CL','Chile');
2
3  COPY paises FROM 'path2foo.csv' DELIMITER ',' CSV;
4
5  UPDATE ciudades SET codigo_postal='81121' WHERE nombre = 'Anton';
6
7  DELETE FROM ciudades WHERE nombre = 'Anton';
8
9  ALTER TABLE delivery ADD COLUMN event_time TIMESTAMP;
10
11 ALTER TABLE ciudades ADD COLUMN num_packages INTEGER DEFAULT 0;
```

CRUD

Create, Read, Update, Delete

```
1  INSERT INTO paises VALUES ('US', 'United States'),('CL','Chile');
2
3  COPY paises FROM 'path2foo.csv' DELIMITER ',' CSV;
4
5  UPDATE ciudades SET codigo_postal='81121' WHERE nombre = 'Anton';
6
7  DELETE FROM ciudades WHERE nombre = 'Anton';
8
9  ALTER TABLE delivery ADD COLUMN event_time TIMESTAMP;
10
11 ALTER TABLE ciudades ADD COLUMN num_packages INTEGER DEFAULT 0;
```

→ INSERT dos tuplas

→ BULK copy desde CSV

→ UPDATE/DELETE a toda
tupla que cumpla condición

→ Agrega un atributo a delivery

→ Agrega un atributo con
un valor por defecto

Condicionales

```
SELECT [ALL/DISTINCT] column_list  
FROM table_list  
[WHERE conditional expression]  
[GROUP BY group_by_column_list]  
[HAVING conditional expression]  
[ORDER BY order_by_column_list]
```

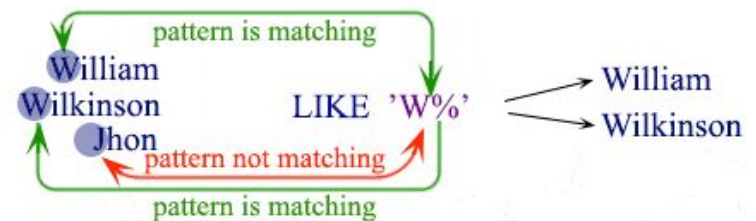
Expresiones condicionales

Syntax : LIKE pattern

Example :

William
Wilkinson
Jhon

LIKE 'W%'

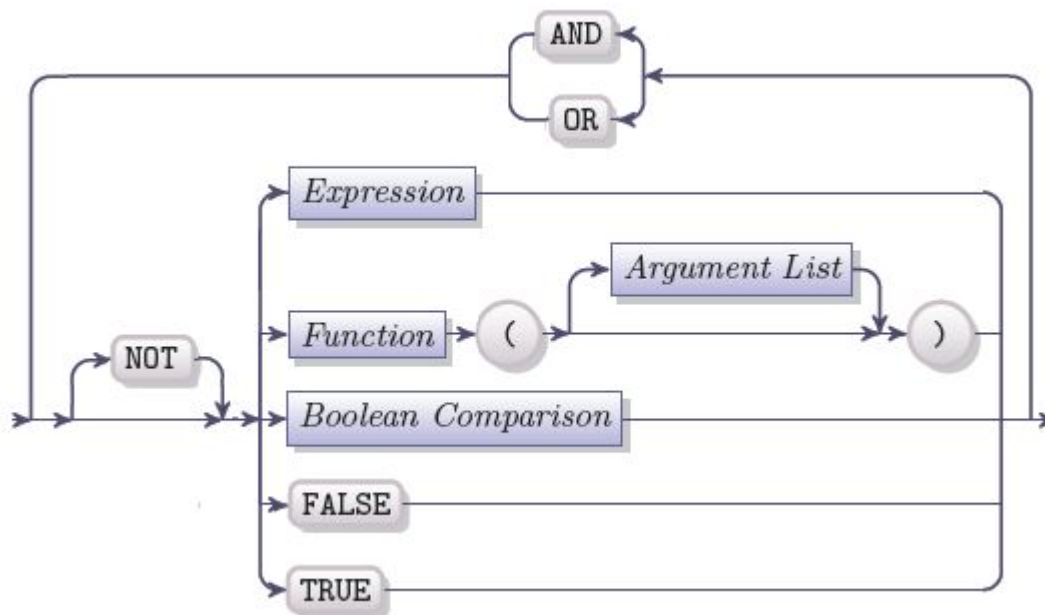


Output :
William
Wilkinson

Condicionales

```
SELECT [ALL/DISTINCT] column_list
FROM table_list
[WHERE conditional expression]
[GROUP BY group_by_column_list]
[HAVING conditional expression]
[ORDER BY order_by_column_list]
```

Expresiones condicionales

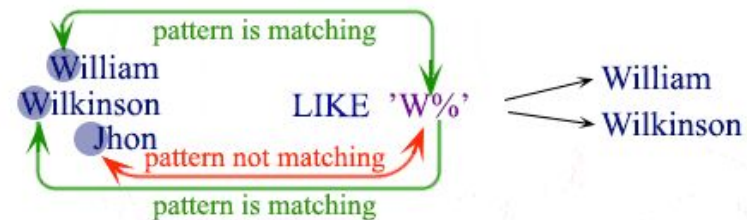


Syntax : LIKE pattern

Example :

William
Wilkinson
Jhon

LIKE 'W%'



Output :
William
Wilkinson

Uso de Join

Operación que chequea condición entre dos relaciones para buscar una coincidencia de valores

Table 1

| | | |
|---|--|--|
| | | |
| 1 | | |
| 2 | | |

Table 2

| | | |
|---|--|--|
| | | |
| 1 | | |
| 3 | | |
| 4 | | |

Outer Join

| | | | |
|---|--|--|--|
| | | | |
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |

Inner Join

| | | | |
|---|--|--|--|
| | | | |
| 1 | | | |

Left Join

| | | | |
|---|--|--|--|
| | | | |
| 1 | | | |
| 2 | | | |

Union

| | | |
|---|--|--|
| | | |
| 1 | | |
| 2 | | |
| 1 | | |
| 3 | | |
| 4 | | |

Cross Join

| | | | |
|---|--|---|--|
| | | | |
| 1 | | 1 | |
| 1 | | 3 | |
| 1 | | 4 | |
| 2 | | 1 | |
| 2 | | 3 | |
| 2 | | 4 | |

```
1 select ciudades.*, paises.nombre
2 from ciudades inner join paises on ciudades.codigo = paises.codigo;
```


Uso de Join

```
1 select ciudades.*, paises.nombre
2 from ciudades inner join paises on ciudades.codigo = paises.codigo;
```

↓ explain analyze

| QUERY PLAN |
|---|
| Hash Join (cost=7.49..1211.70 rows=40135 width=29) (actual time=1.951..155.426 rows=40135 loops=1) |
| Hash Cond: (ciudades.codigo = paises.codigo) |
| -> Seq Scan on ciudades (cost=0.00..652.35 rows=40135 width=19) (actual time=0.035..58.370 rows=40135 loops=1) |
| -> Hash (cost=4.44..4.44 rows=244 width=13) (actual time=1.816..1.816 rows=244 loops=1) |
| Buckets: 1024 Batches: 1 Memory Usage: 11kB |
| -> Seq Scan on paises (cost=0.00..4.44 rows=244 width=13) (actual time=0.305..0.834 rows=244 loops=1) |
| Total runtime: 164.851 ms |

Query - subquery

```
SELECT DISTINCT CustomerID  
FROM Sales.SalesOrderHeader  
WHERE TerritoryID IN (SELECT TerritoryID  
FROM Sales.SalesTerritory  
WHERE SalesYTD < 5000000)
```

Subquery



Query - subquery

```
SELECT DISTINCT CustomerID  
FROM Sales.SalesOrderHeader  
WHERE TerritoryID IN (SELECT TerritoryID  
FROM Sales.SalesTerritory  
WHERE SalesYTD < 5000000)
```

Subquery

```
1 select e.nombre, e.codigo_postal  
2 from ciudades e  
3 where e.codigo_postal in  
4     (select codigo_postal  
5      from ciudades  
6      where nombre like 'An%');
```

Query - subquery

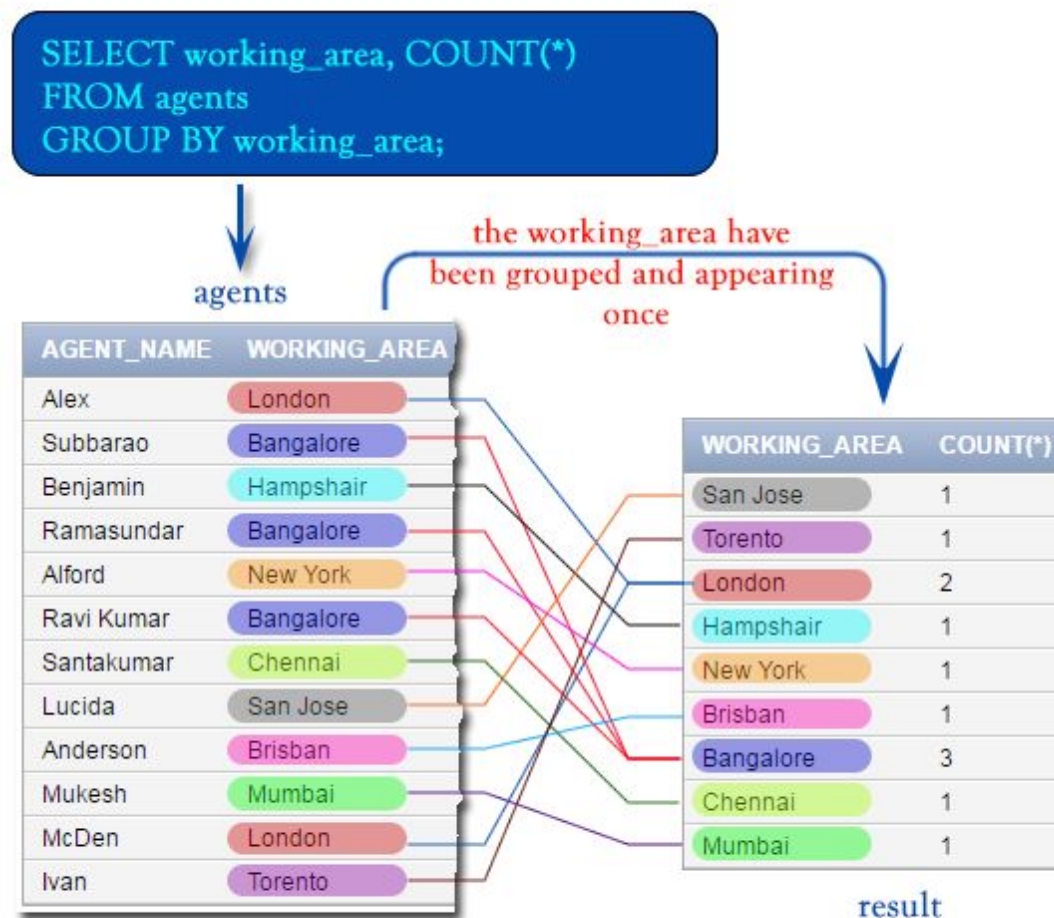
```
1 select e.nombre, e.codigo_postal
2 from ciudades e
3 where e.codigo_postal in
4       (select codigo_postal
5        from ciudades
6        where nombre like 'An%');
```

explain analyze

| QUERY PLAN |
|--|
| Hash Join (cost=243.46..1050.90 rows=1904 width=16) (actual time=2.602..41.791 rows=456 loops=1) |
| Hash Cond: ((e.codigo_postal)::text = (ciudades.codigo_postal)::text) |
| -> Seq Scan on ciudades e (cost=0.00..652.35 rows=40135 width=16) (actual time=0.057..18.832 rows=40135 loops=1) |
| -> Hash (cost=240.97..240.97 rows=199 width=6) (actual time=2.492..2.492 rows=194 loops=1) |
| Buckets: 1024 Batches: 1 Memory Usage: 8kB |
| -> HashAggregate (cost=238.98..240.97 rows=199 width=6) (actual time=2.163..2.272 rows=194 loops=1) |
| -> Bitmap Heap Scan on ciudades (cost=5.90..238.03 rows=380 width=6) (actual time=0.496..1.846 rows=207 loops=1) |
| Filter: (nombre ~~ 'An%':text) |
| -> Bitmap Index Scan on ciudades_pkey (cost=0.00..5.81 rows=155 width=0) (actual time=0.452..0.452 rows=207 loops=1) |
| Index Cond: ((nombre >= 'An':text) AND (nombre < 'Ao':text)) |
| Total runtime: 42.386 ms |

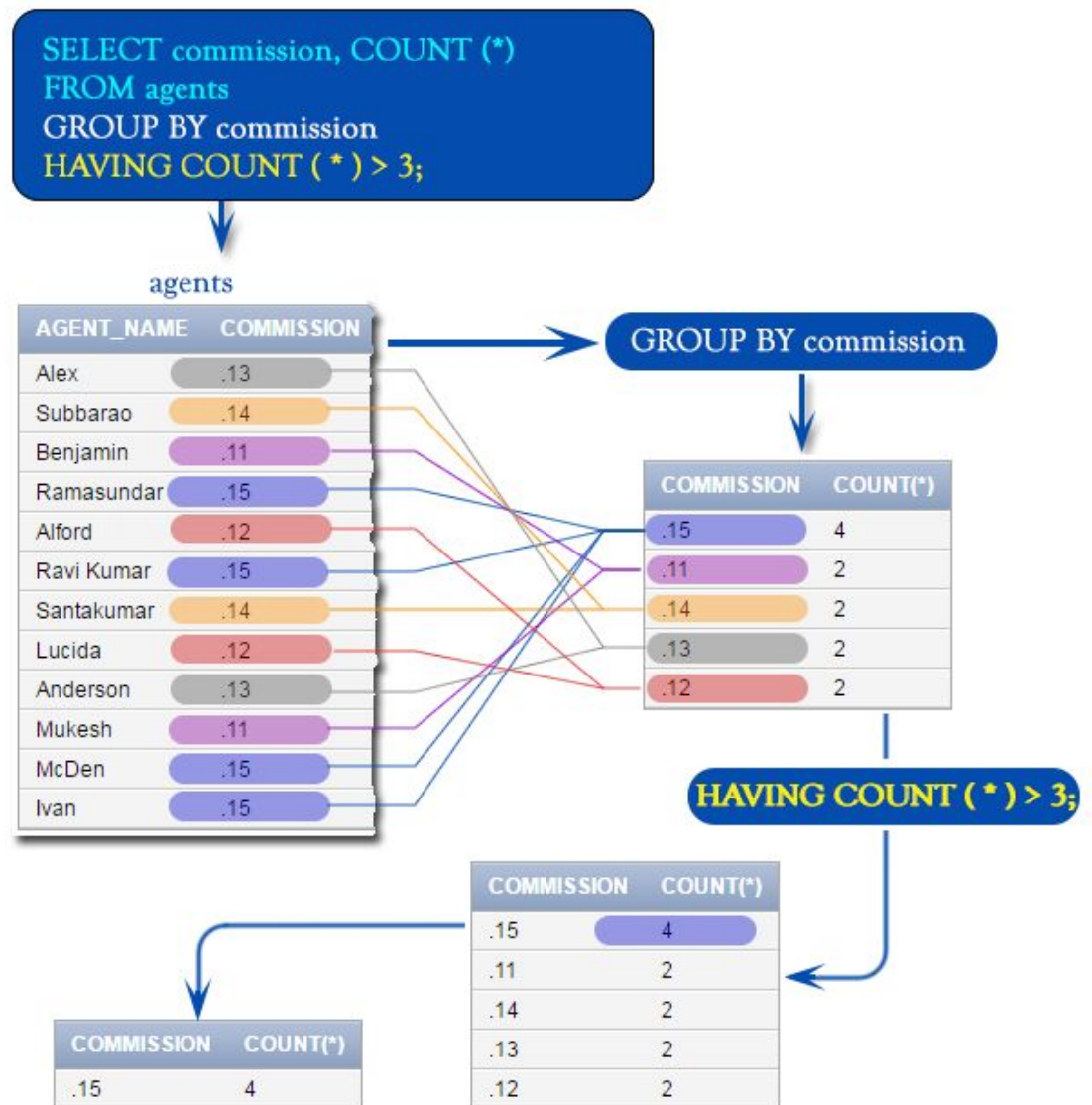
Group by

Chequeo entre grupos de registros en torno a la coincidencia de un valor



Having

Cláusula que permite evaluar una condición sobre cada uno de los grupos generados por **GROUP BY**



Vista

Se calculan **en la medida que sean requeridas** - no dinámicas


```
1 create view paises_ciudades as
2 select ciudades.nombre as ciudad, paises.nombre as pais
3 from ciudades, paises where ciudades.codigo = paises.codigo;
4
5 create table mat_view as
6 select ciudades.nombre as ciudad, paises.nombre as pais
7 from ciudades, paises where ciudades.codigo = paises.codigo;
```

Vista

Se calculan **en la medida que sean requeridas** - no dinámicas

```
1 create view paises_ciudades as
2 select ciudades.nombre as ciudad, paises.nombre as pais
3 from ciudades, paises where ciudades.codigo = paises.codigo;
4
5 create table mat_view as
6 select ciudades.nombre as ciudad, paises.nombre as pais
7 from ciudades, paises where ciudades.codigo = paises.codigo;
```

```
explain analyze select * from paises_ciudades where pais = 'USA';
```




Vista

Se calculan **en la medida que sean requeridas** - no dinámicas

```
1 create view paises_ciudades as
2 select ciudades.nombre as ciudad, paises.nombre as pais
3 from ciudades, paises where ciudades.codigo = paises.codigo;
4
5 create table mat_view as
6 select ciudades.nombre as ciudad, paises.nombre as pais
7 from ciudades, paises where ciudades.codigo = paises.codigo;
```

```
explain analyze select * from paises_ciudades where pais = 'USA';
```



| QUERY PLAN |
|--|
| Hash Join (cost=5.06..809.56 rows=164 width=20) (actual time=0.795..159.705 rows=40135 loops=1) |
| Hash Cond: (ciudades.codigo = paises.codigo) |
| -> Seq Scan on ciudades (cost=0.00..652.35 rows=40135 width=13) (actual time=0.097..80.523 rows=40135 loops=1) |
| -> Hash (cost=5.05..5.05 rows=1 width=13) (actual time=0.259..0.259 rows=1 loops=1) |
| Buckets: 1024 Batches: 1 Memory Usage: 1kB |
| -> Seq Scan on paises (cost=0.00..5.05 rows=1 width=13) (actual time=0.106..0.248 rows=1 loops=1) |
| Filter: (nombre = 'USA'::text) |
| Rows Removed by Filter: 243 |
| Total runtime: 167.153 ms |

Vista materializada

Se actualiza constantemente, basadas **en relación**

```
1 create view paises_ciudades as
2 select ciudades.nombre as ciudad, paises.nombre as pais
3 from ciudades, paises where ciudades.codigo = paises.codigo;
4
5 create table mat_view as
6 select ciudades.nombre as ciudad, paises.nombre as pais
7 from ciudades, paises where ciudades.codigo = paises.codigo;
```

```
explain analyze select ciudades.nombre, paises.nombre
from ciudades, paises where ciudades.codigo = paises.codigo and paises.nombre = 'USA';
```

QUERY PLAN

Hash Join (cost=5.06..809.56 rows=164 width=20) (actual time=0.434..79.118 rows=40135 loops=1)

Hash Cond: (ciudades.codigo = paises.codigo)

-> Seq Scan on ciudades (cost=0.00..652.35 rows=40135 width=13) (actual time=0.072..22.009 rows=40135 loops=1)

-> Hash (cost=5.05..5.05 rows=1 width=13) (actual time=0.278..0.278 rows=1 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 1kB

-> Seq Scan on paises (cost=0.00..5.05 rows=1 width=13) (actual time=0.072..0.268 rows=1 loops=1)

Filter: (nombre = 'USA'::text)

Rows Removed by Filter: 243

Total runtime: 84.240 ms

Trigger

```
1 CREATE OR REPLACE FUNCTION packages_inc() RETURNS TRIGGER AS $$
2 DECLARE myrec RECORD;
3 BEGIN
4 SELECT * INTO myrec FROM delivery
5 WHERE delivery.event_time = NEW.event_time;
6 UPDATE ciudades SET num_packages = num_packages + 1
7 WHERE codigo_postal = myrec.codigo_llegada;
8 RETURN NEW;
9 END;
10 $$ LANGUAGE plpgsql;
11
12 CREATE TRIGGER package_trigger AFTER INSERT ON delivery
13 FOR EACH ROW EXECUTE PROCEDURE packages_inc();
```

Trigger

```
1 CREATE OR REPLACE FUNCTION packages_inc() RETURNS TRIGGER AS $$
2 DECLARE myrec RECORD;
3 BEGIN
4 SELECT * INTO myrec FROM delivery
5 WHERE delivery.event_time = NEW.event_time;
6 UPDATE ciudades SET num_packages = num_packages + 1
7 WHERE codigo_postal = myrec.codigo_llegada;
8 RETURN NEW;
9 END;
10 $$ LANGUAGE plpgsql;
11
12 CREATE TRIGGER package_trigger AFTER INSERT ON delivery
13 FOR EACH ROW EXECUTE PROCEDURE packages_inc();
```

INSERT, UPDATE, DELETE

```
[ DECLARE
    declarations ]
BEGIN
    statements
END;
```

```
CREATE [ CONSTRAINT ] TRIGGER name { BEFORE | AFTER | INSTEAD OF } { event [ OR ... ] }
ON table_name
[ FROM referenced_table_name ]
[ NOT DEFERRABLE | [ DEFERRABLE ] { INITIALLY IMMEDIATE | INITIALLY DEFERRED } ]
[ FOR [ EACH ] { ROW | STATEMENT } ]
[ WHEN ( condition ) ]
EXECUTE PROCEDURE function_name ( arguments )
```

Índice

Estructura persistente que permite acceso eficiente cuando las búsqueda se realiza en base a un atributo no clave

```
CREATE INDEX ind_nombre ON paises USING HASH (nombre);
```

```
CREATE INDEX ind_nombre ON paises USING BTREE (nombre);
```

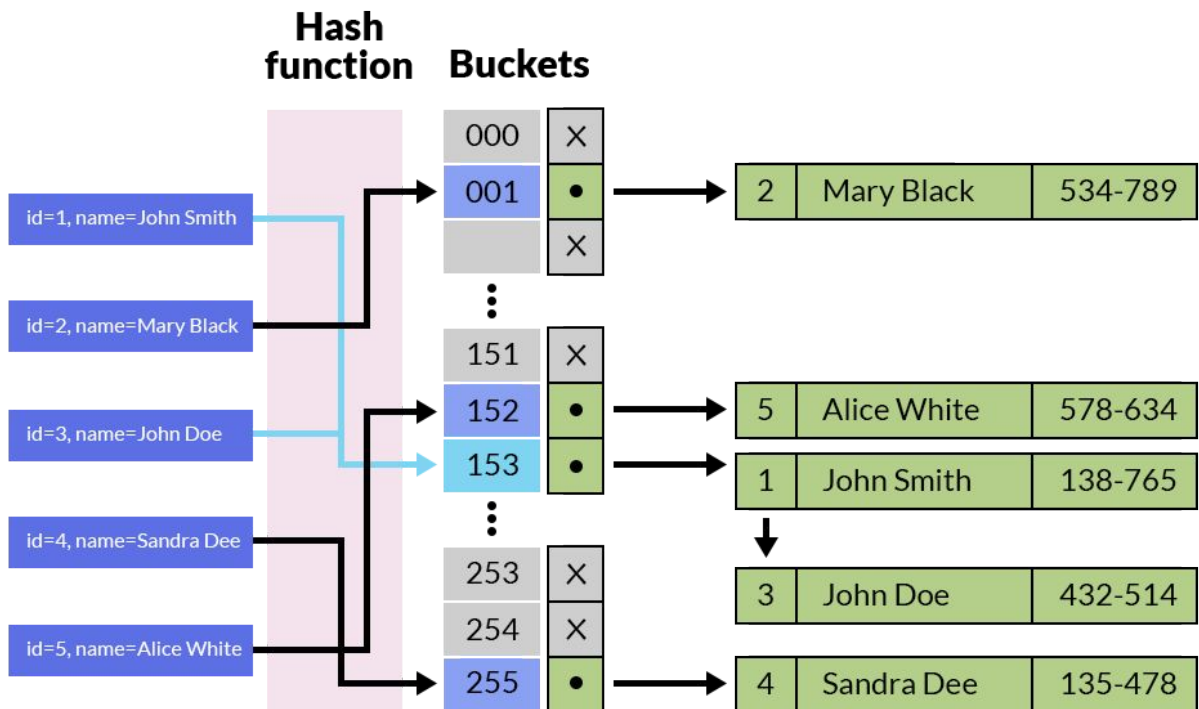

Índice

Estructura persistente que permite acceso eficiente cuando las búsqueda se realiza en base a un atributo no clave

```
CREATE INDEX ind_nombre ON países USING HASH (nombre);
```

```
CREATE INDEX ind_nombre ON países USING BTREE (nombre);
```

valores



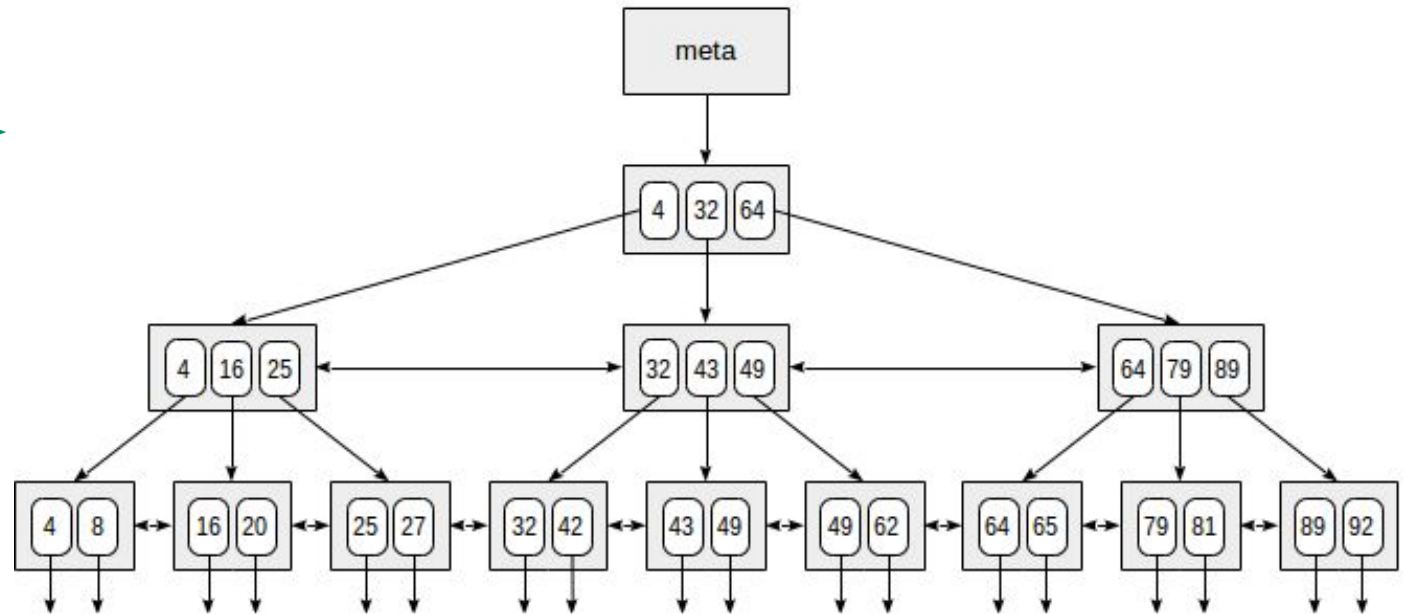
Índice

Estructura persistente que permite acceso eficiente cuando las búsqueda se realiza en base a un atributo no clave

```
CREATE INDEX ind_nombre ON países USING HASH (nombre);
```

```
CREATE INDEX ind_nombre ON países USING BTREE (nombre);
```

intervalo



Consultas?

Recuerden!

- Marcelo Mendoza: mmendoza@inf.utfsm.cl
- Margarita Bugueño: margarita.bugueno@usm.cl