

Laboratorio 6

Integrantes:

- Gino Jesús Daza Yalta
- Mikel Dan Bracamonte Toguchi

P1. (2 puntos) Escaneo Secuencial vs Índice GIN

1. Crear tabla:

```
CREATE EXTENSION IF NOT EXISTS pg_trgm;  
CREATE TABLE articles (  
    content_lineal text,  
    content_gin text  
)
```

2. Insertar datos aleatorios:

```
-- Inserta 10^7 registros aleatorios  
INSERT INTO articles (content_lineal)  
SELECT md5(random()::text)  
FROM generate_series(1, 10000000) AS id;  
  
-- Copia a la columna indexada  
UPDATE articles SET content_gin = content_lineal;
```

3. Crear índice GIN con trigramas:

```
CREATE INDEX idx_articles_trgm_gin ON articles  
USING GIN (content_gin gin_trgm_ops);
```

4. Crear subconjuntos más pequeños:

```
CREATE TABLE articles_1k AS SELECT * FROM articles LIMIT 1000;  
CREATE TABLE articles_10k AS SELECT * FROM articles LIMIT 10000;  
CREATE TABLE articles_100k AS SELECT * FROM articles LIMIT 100000;  
CREATE TABLE articles_1m AS SELECT * FROM articles LIMIT 1000000;
```

5. Crear índice en las copias:

```
CREATE INDEX idx_trgm_gin_1k ON articles_1k USING GIN (content_gin gin_trgm_ops);  
CREATE INDEX idx_trgm_gin_10k ON articles_10k USING GIN (content_gin  
gin_trgm_ops);  
CREATE INDEX idx_trgm_gin_100k ON articles_100k USING GIN (content_gin  
gin_trgm_ops);
```

```
CREATE INDEX idx_trgm_gin_1m ON articles_1m USING GIN (content_gin
gin_trgm_ops);
```

6. Consultas sin y con índice:

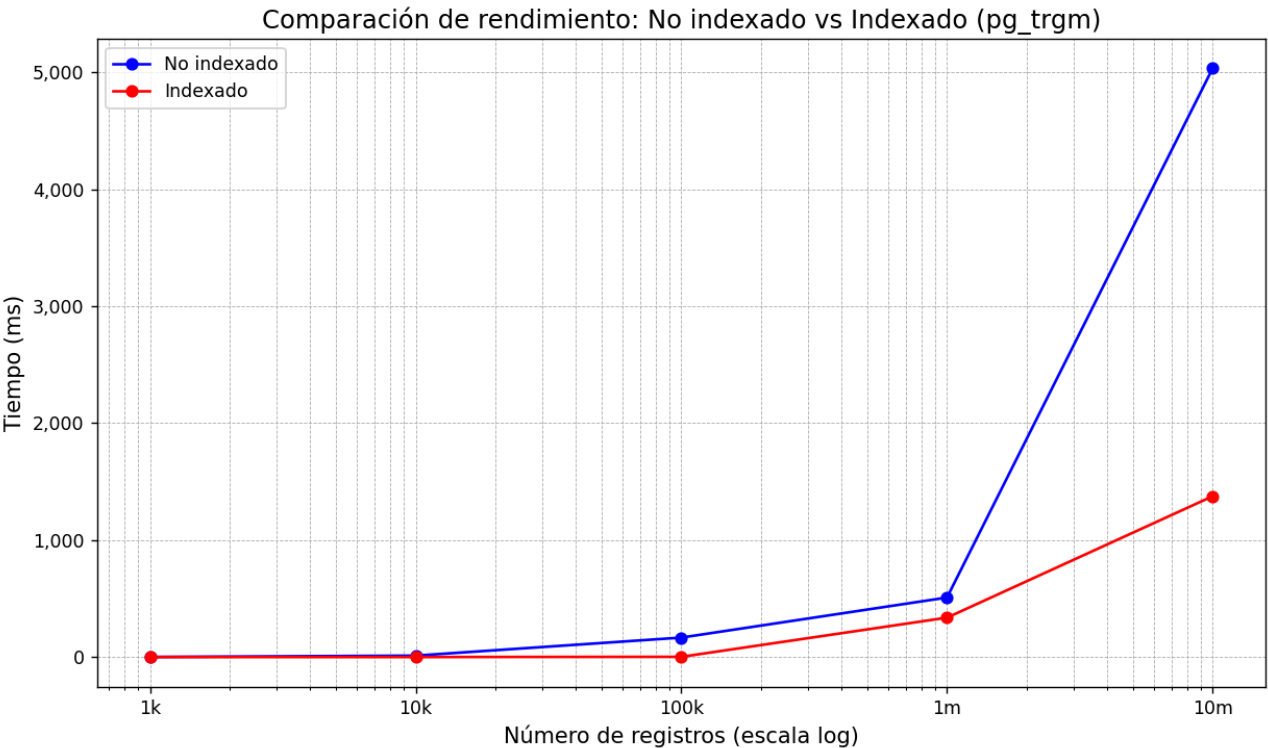
Sin indice:

```
EXPLAIN ANALYZE
SELECT COUNT(*) FROM articles_10k
WHERE content_lineal ILIKE '%abc%';
```

Con indice:

```
EXPLAIN ANALYZE
SELECT COUNT(*) FROM articles_10k
WHERE content_gin ILIKE '%abc%';
```

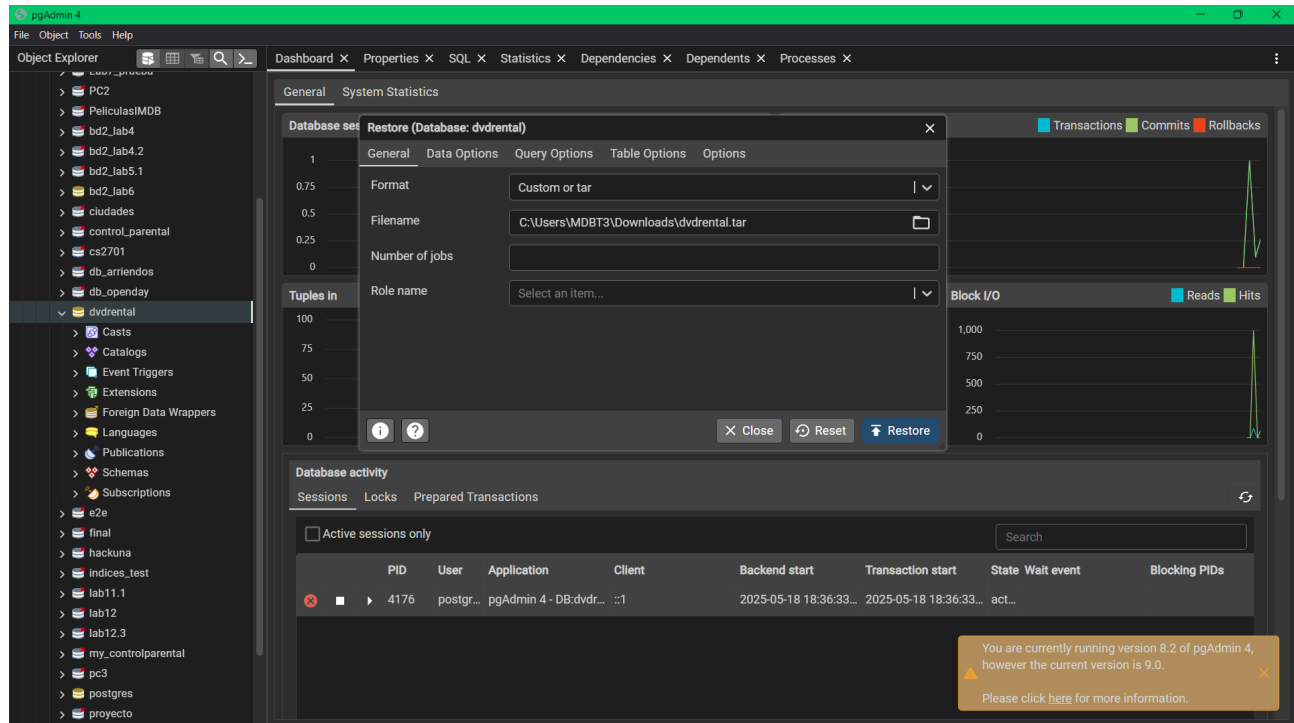
Cantidad de datos	Sin indice (ms)	Con indice (ms)
1k	1.23	0.094
10k	12.428	0.377
100k	166.779	2.298
1m	509.202	337.237
10m	5033.666	1375.336



El gráfico evidencia que el índice GIN basado en trigramas (pg_trgm) reduce significativamente los tiempos de búsqueda a medida que crece el volumen de datos. Mientras que sin índice el tiempo escala rápidamente (hasta 5 segundos en 10 millones de registros), el uso del índice permite mantener los tiempos bajo control, incluso en escalas grandes.

P2. (6 puntos) Búsqueda de Texto Completo en Películas

Primero, se restauró la base de datos con la herramienta de pgadmin.



Luego se crearon dos nuevas columnas donde se concatenaron el título y descripción en un vector de pesos. A una columna se le creó un índice GIN.

```
ALTER TABLE film ADD COLUMN vector_lineal tsvector;
```

```
ALTER TABLE film ADD COLUMN vector_gin tsvector;
```

```
UPDATE film SET vector_lineal = to_tsvector('english', title || description);
```

```
UPDATE film SET vector_gin = vector_lineal;
```

```
CREATE INDEX idx_film_vector ON film USING GIN(vector_gin); --usa keywords
```

Se hicieron consultas con y sin el índice mediante las siguientes queries y se midieron los tiempos:

```
EXPLAIN ANALYZE
```

```
SELECT title, description, vector_gin FROM film WHERE vector_gin @@  
to_tsquery('english', 'Man & Woman')
```

```
EXPLAIN ANALYZE
```

SELECT title, description, vector_lineal FROM film WHERE vector_lineal @@
to_tsquery('english', 'Man & Woman')

Luego se duplicaron los registros mediante la siguiente instrucción, y se volvieron a medir los tiempos

INSERT INTO film (title, description, language_id, vector_lineal, vector_gin) SELECT title,
description, language_id, vector_lineal, vector_gin FROM film

Los resultados fueron los siguientes:

Cantidad de registros	Tiempo sin índice (ms)	Tiempo con índice(ms)
1000	0.538	0.153
2000	0.762	0.157
4000	1.376	0.161
8000	2.643	0.223
16000	6.204	0.422

Por ejemplo para 1000 registros:
Con índice:

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer showing a tree of database objects, with 'dvdrental' selected. The main pane displays a SQL query in the 'Query' tab:

```
1 EXPLAIN ANALYZE
2 SELECT title, description, vector_gin FROM film WHERE vector_gin @@ to_tsquery('english', 'Man & Woman')
3
4 EXPLAIN ANALYZE
5 SELECT title, description, vector_lineal FROM film WHERE vector_lineal @@ to_tsquery('english', 'Man & Woman')
6
```

Below the query, the 'Data Output' tab shows the 'QUERY PLAN' for the second query:

```
QUERY PLAN
text
1  Bitmap Heap Scan on film (cost=21.57..73.14 rows=16 width=287) (actual time=0.082..0.116 rows=15 loops=1)
2    Recheck Cond: (vector_gin @@ "man" & "woman":tsquery)
3    Heap Blocks: exact=14
4    -> Bitmap Index Scan on idx_film_vector (cost=0.00..21.57 rows=16 width=0) (actual time=0.072..0.073 rows=15 loops=1)
5          Index Cond: (vector_gin @@ "man" & "woman":tsquery)
6  Planning Time: 0.247 ms
7  Execution Time: 0.153 ms
```

At the bottom, it indicates 'Total rows: 7 of 7' and 'Query complete 00:00:00.058'.

Acá se puede observar que sí se está usando el índice, ya que se hace un bitmap index scan usando el índice.

Sin índice:

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer with a tree view of database objects. The main pane displays a SQL query in the Query editor:

```
1 EXPLAIN ANALYZE
2 SELECT title, description, vector_gin FROM film WHERE vector_gin @@ to_tsquery('english', 'Man & Woman')
3
4 EXPLAIN ANALYZE
5 SELECT title, description, vector_linalg FROM film WHERE vector_linalg @@ to_tsquery('english', 'Man & Woman')
6
```

Below the query editor, the Data Output tab shows the query plan:

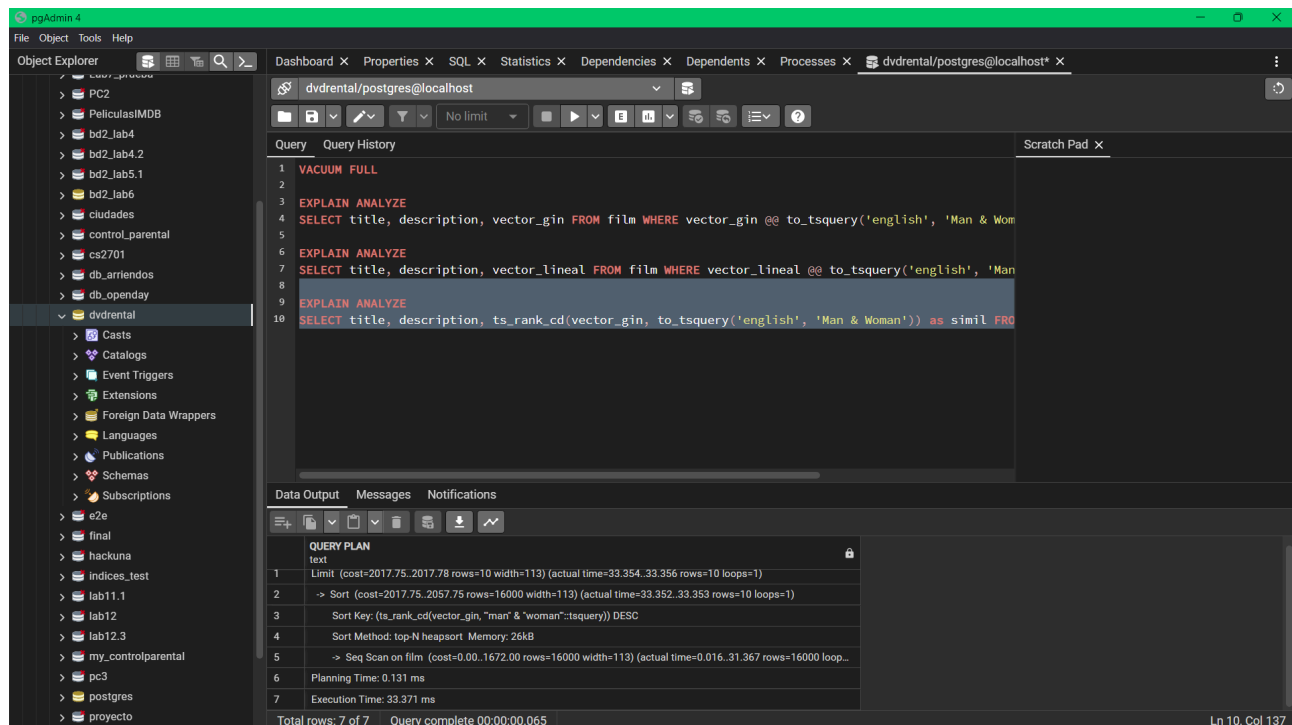
QUERY PLAN	
1	Seq Scan on film (cost=0.00..243.50 rows=16 width=287) (actual time=0.118..0.529 rows=15 loops=1)
2	Filter: (vector_linalg @@ 'man' & 'woman')::tsquery)
3	Rows Removed by Filter: 985
4	Planning Time: 0.198 ms
5	Execution Time: 0.538 ms

At the bottom, it shows 'Total rows: 5 of 5' and 'Query complete 00:00:00.041'.

Luego, se hizo la misma consulta (ahora hay 16000 datos), pero se rankearon por orden de similitud mediante la siguiente query:

```
SELECT title, description, ts_rank_cd(vector_gin, to_tsquery('english', 'Man & Woman')) as  
simil FROM film order by simil desc limit 10;
```

Resultado:



El tiempo de ejecución fue de 33.371ms, mucho más que el tiempo anterior sin rankear de 0.422ms. Además, se puede observar que el índice no se está usando, y se está haciendo un sequential scan.

P3. (6 puntos) Búsqueda de Texto Completo en Noticias

1. Crear la tabla articles e importar csv (se hizo con el import de pgadmin):

```

CREATE TABLE articles(
    num INT PRIMARY KEY,
    id INT,
    title TEXT,
    publication TEXT,
    author TEXT,
    date DATE,
    year FLOAT,
    month FLOAT,
    url TEXT,
    content TEXT
)

```

2. Generar un nuevo atributo indexado que combine el título y el contenido de cada noticia:

```

ALTER TABLE articles ADD COLUMN full_text_idx tsvector;

UPDATE articles

```

```
SET full_text_idx = to_tsvector('english', coalesce(title, '') || ' ' || coalesce(content, ''));
```

3. Crear un índice GIN sobre ese nuevo atributo:

```
CREATE INDEX gin_articles_idx ON articles USING GIN (full_text_idx);
```

4. Ejecutar consultas de texto completo:

4.1 Resultados sin índice GIN:

```
EXPLAIN ANALYZE
SELECT * FROM articles
WHERE to_tsvector('english', title || ' ' || content) @@ plainto_tsquery('english', 'trump');
```

Query

Query History

1

SELECT * FROM articles

2

WHERE to_tsvector('english', title || ' ' || content) @@ plainto_tsquery('english', 'trump');

3

Data Output

Messages

Notifications

SQL

Showing rows: 1 to 1000

Page No: 1

of 11

	num [PK] integer	id integer	title text	publicat text	author text	date date	year double	month double	url text	content text
1	30993	49758	Wikileaks: John Harwood Boasted of Goadng Trump at Debate	Brei...	Joel ...	2016-10-11	2016	10	[null]	John Harwood of CNBC and The New York Times, who moderated one of the early Republican pr
2	30997	49762	Supreme Court Punts on Little Sisters' Obamacare Case Until After 2...	Brei...	Ken ...	2016-05-16	2016	5	[null]	WASHINGTON — "The Court expresses no view on the merits of the cases. "[With that, the Supre
3	31024	49789	10 Violent Actions Against Trump Supporters - Breitbart	Brei...	John ...	2016-11-17	2016	11	[null]	It looks like an awful lot of Hillary Clinton supporters didn't receive her campaign messages aboi
4	31025	49790	TSA's Head of Security Forced Out over Airport Delays - Breitbart	Brei...	John ...	2016-05-24	2016	5	[null]	Kelly Hoggan, head of security operations for the Transportation Security Administration, has bei
5	31026	49791	Marine Corps Times Urges Veterans To Support More Gun Control - B...	Brei...	AWR ...	2016-01-18	2016	1	[null]	On January 18, the Marine Corps Times criticized the " attitude toward gun control" and urged v
6	31028	49793	Obama Administration Nearly Doubles Number of Refugee Arrivals S...	Brei...	Mich...	2016-12-26	2016	12	[null]	The Obama administration has accepted 25,584 refugees into the United States in the two mont
7	31031	49796	Pamela Geller: SPLC Publishes Libelous New Hit List of Anti-Jihad Vo...	Brei...	Pam...	2016-10-27	2016	10	[null]	As the jihad threat grows, so does the enemies' war on our most effective leaders. The SPLC, tra
8	31076	49841	Marc Rich Pardon Still Paying Off for Bill Clinton	Brei...	Breit...	2016-01-17	2016	1	[null]	Peter Schweizer — for Breitbart News, President of the Government Accountability Institute, ar
9	31113	49878	Ryan Challenger Responds to #NeverTrump Movement with #NeverR...	Brei...	Julia ...	2016-04-05	2016	4	[null]	House Speaker Paul Ryan's primary challenger, Wisconsin businessman Paul Nehlen, responded
10	31135	49900	Personal and Financial Scandal Follow Rubio into Tuesday Florida Vo...	Brei...	Julia ...	2016-03-12	2016	3	[null]	MIAMI, FL — Ahead of Florida's Tuesday primary, new attention is being paid to Rubio's persona
11	31149	49914	Limbaugh: Trump Built the 'Coalition' the GOP Claims to Want - Now ...	Brei...	Jeff ...	2016-02-10	2016	2	[null]	Wednesday on his radio show, conservative talk show host Rush Limbaugh argued that New Han
12	31159	49924	Judge: Texas Voter ID Law Will Be Enforced this November	Brei...	Loga...	2016-07-22	2016	7	[null]	Shortly after an appeals court ruled Texas' voter identification law to have discriminatory effects
13	31160	49925	Jeff Sessions: Obama Must 'Make Crystal Clear' Only Citizens Can V...	Brei...	Katie...	2016-11-07	2016	11	[null]	Alabama Republican Sen. Jeff Sessions is condemning President Obama's refusal to state that il
14	31196	49961	Supreme Court Upholds Terror Victims Judgment Against Iran	Brei...	Edwi...	2016-04-21	2016	4	[null]	Families of the victims of the 1983 Marine barracks bombing in Beirut and other terrorist attacks
15	5784	23832	As Donald Trump Repels Minority Voters, G.O.P. Fears Its Future in th...	New...	Jere...	2017-02-02	2017	2	[null]	PHOENIX — Republicans in Western states fear that Donald J. Trump could imperil their party f
16	5785	23833	Trump a Working-Class Hero? A Blue-Collar Town Debates His Crede...	New...	Richa...	2017-02-04	2017	2	[null]	YOUNGSTOWN, Ohio — "Is this idiot for Trump?" Mark Wasko asked, his voice booming across
17	5789	23837	Court Overturns 'Burkini' Ban in French Town - The New York Times	New...	Aurel...	2017-02-02	2017	2	[null]	PARIS — France's highest administrative court on Friday overturned a town's ban on burkinis, th
18	5032	22874	Muted U.S. Response to China's Seizure of Drone Worries Asian Allie...	New...	Jane ...	2017-04-13	2017	4	[null]	BEIJING — Only a day before a small Chinese boat sidled up to a United States Navy research v
19	5033	22875	China Agrees to Return Seized Drone, Ending Standoff, Pentagon Say...	New...	Jane ...	2017-04-11	2017	4	[null]	BEIJING — The Pentagon on Saturday said that Beijing had agreed to return an underwater droi
20	5042	22886	As Trump Signals Climate Action Pullback, Local Leaders Push Forw...	New...	Tatia...	2016-12-22	2016	12	[null]	The incoming Trump administration appears determined to reverse much of what President Ob
21	5112	22964	Refugees Encounter a Foreign Word: Welcome - The New York Times	New...	Jodi ...	2016-12-24	2016	12	[null]	TORONTO — One frigid day in February, Kerry McLogg drove to an airport hotel here to pick up a

Total rows: 10096


Query complete 00:00:23.903

CRLF

Ln 3, Col 1

No usa indice:

Prueba de que usa el índice:

Query		Query History
1	EXPLAIN ANALYZE	
2	SELECT * FROM articles	
3	WHERE full_text_idx @@ plainto_tsquery('english', 'trump');	
Data Output		Messages Notifications
		
QUERY PLAN		
text		
1	Bitmap Heap Scan on articles (cost=150.22..10443.19 rows=19857 width=796) (actual time=4.007..75.819 rows=19907 loops=1)	
2	Recheck Cond: (full_text_idx @@ "trump"::tsquery)	
3	Heap Blocks: exact=6706	
4	-> Bitmap Index Scan on gin_articles_idx (cost=0.00..145.25 rows=19857 width=0) (actual time=2.358..2.359 rows=19907 loop...)	
5	Index Cond: (full_text_idx @@ "trump"::tsquery)	
6	Planning Time: 0.166 ms	
7	Execution Time: 77.334 ms	

5. Crear subconjuntos de datos aleatorios:

```
CREATE TABLE articles_1k AS SELECT * FROM articles ORDER BY random() LIMIT 1000;
CREATE TABLE articles_10k AS SELECT * FROM articles ORDER BY random() LIMIT 10000;
CREATE TABLE articles_25k AS SELECT * FROM articles ORDER BY random() LIMIT 25000;
CREATE TABLE articles_50k AS SELECT * FROM articles ORDER BY random() LIMIT 50000;
```

6. Medir los tiempos de ejecución para diferentes tamaños de tabla:

6.1. Se hará la comparación con esta consulta:

Sin indice:

```
SELECT *, ts_rank(to_tsvector('english', title || ' ' || content), plainto_tsquery('english',
'climate change')) AS rank
FROM articles_50k
WHERE to_tsvector('english', title || ' ' || content) @@ plainto_tsquery('english', 'climate
change')
ORDER BY rank DESC
LIMIT 10;
```

Con indice:

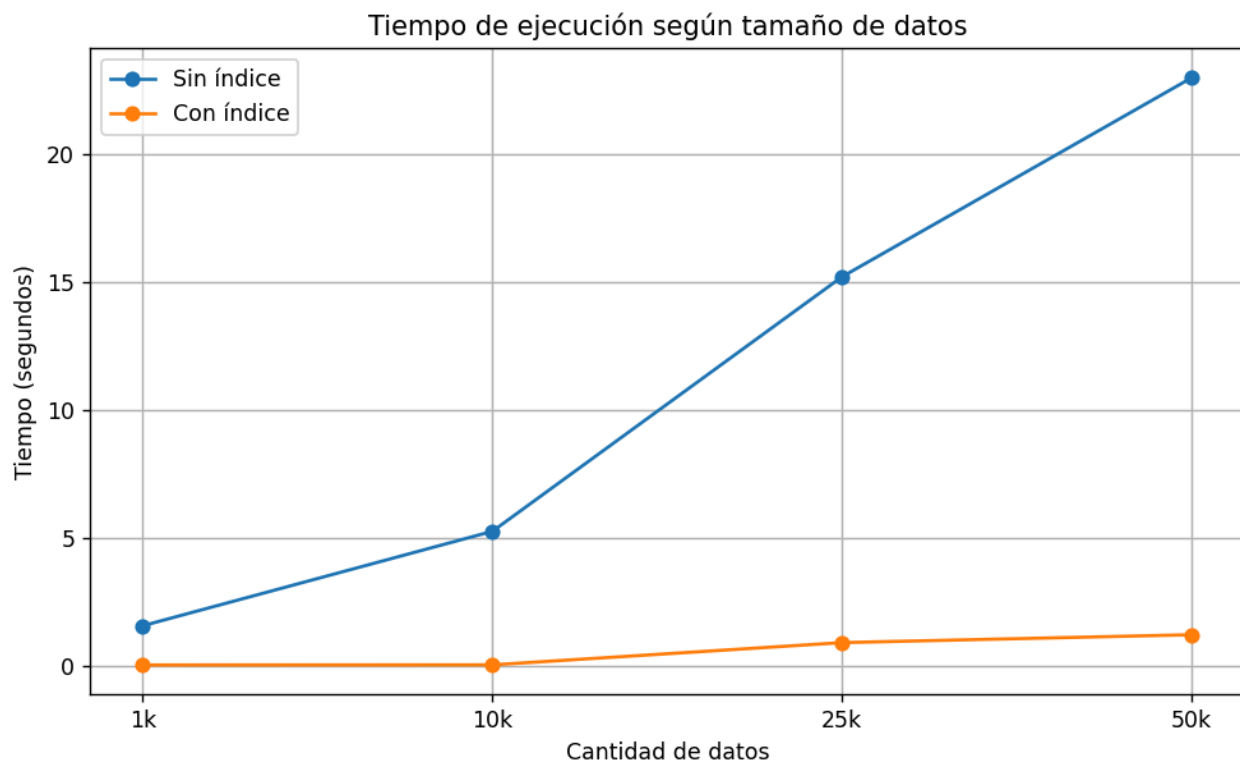
```
SELECT *, ts_rank(full_text_idx, plainto_tsquery('english', 'climate change')) AS rank
FROM articles_50k
WHERE full_text_idx @@ plainto_tsquery('english', 'climate change')
```

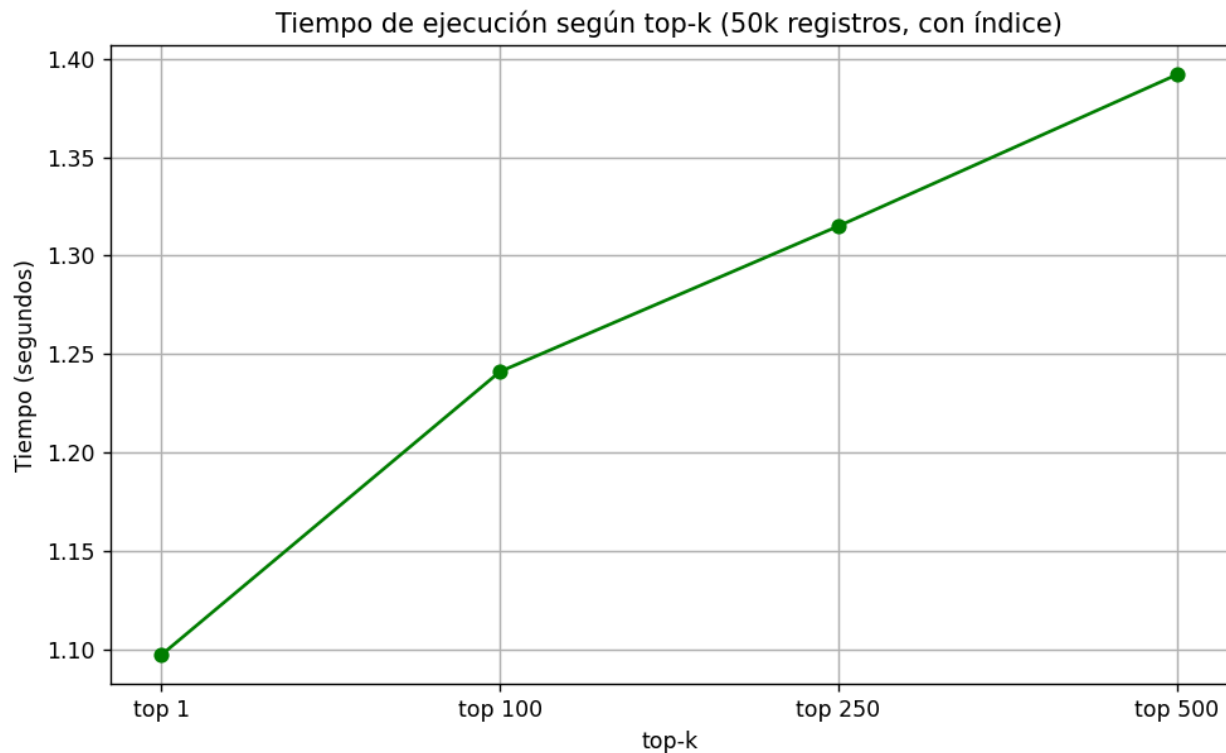
ORDER BY rank DESC
LIMIT 10;

#Datos	Sin indice (seg)	Con indice (seg)
1k	1.553	0.0168
10k	5.250	0.0217
25k	15.213	0.892
50k	23.004	1.201

50k datos, con indice, top k:	Tiempo
top 1	1.097
top 100	1.241
top 250	1.315
top 500	1.392

Gráficos comparativos:





El primer gráfico muestra que el uso del índice GIN mejora significativamente el rendimiento a medida que crece el volumen de datos. El segundo gráfico demuestra que aumentar el valor de top-k tiene un impacto moderado en el tiempo de ejecución, aunque el índice sigue manteniendo un rendimiento eficiente.

P4. (6 puntos) Comparación de Índices GIN vs GIST en Noticias

Usando el mismo dataset de la pregunta anterior, se crearon dos tablas idénticas.

```
CREATE TABLE articles_gin(  
    num INT PRIMARY KEY,  
    id INT,  
    title TEXT,  
    publication TEXT,  
    author TEXT,  
    date DATE,  
    year FLOAT,  
    month FLOAT,  
    url TEXT,  
    content TEXT  
);
```

```
CREATE TABLE articles_gist(  
    num INT PRIMARY KEY,  
    id INT,
```

```
        title TEXT,  
        publication TEXT,  
        author TEXT,  
        date DATE,  
        year FLOAT,  
        month FLOAT,  
        url TEXT,  
        content TEXT  
    );
```

Se insertaron los datos del dataset usando la herramienta de pgadmin, y luego se crearon dos nuevas columnas donde estará el full_text.

```
ALTER TABLE articles_gin ADD COLUMN full_text_idx tsvector;  
ALTER TABLE articles_gist ADD COLUMN full_text_idx tsvector;  
  
UPDATE articles_gin  
SET full_text_idx = to_tsvector('english', coalesce(title, "") || ' ' || coalesce(content, ""));  
  
INSERT INTO articles_gist SELECT * FROM articles_gin;
```

Ahora tenemos dos tablas idénticas, una se usará para el índice gin y la otra para el índice gist.

Primero, se crearon los índices respectivos usando las siguientes instrucciones y se tomó el tiempo de creación:

```
CREATE INDEX gin_articles_idx ON articles_gin USING GIN (full_text_idx);  
CREATE INDEX gist_articles_idx ON articles_gist USING GIST (full_text_idx);
```

Tiempo de creación:

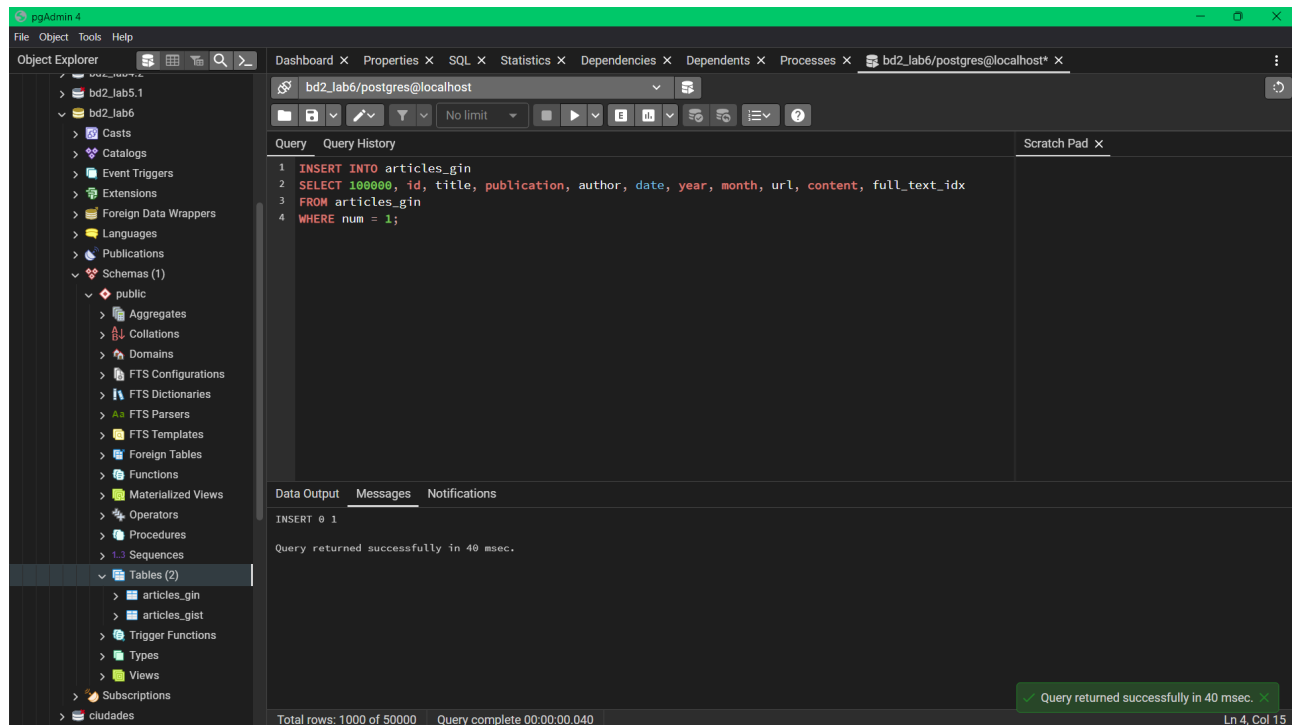
Índice gin:


```
INSERT INTO articles_gin
SELECT 100000, id, title, publication, author, date, year, month, url, content, full_text_idx
FROM articles_gin
WHERE num = 1;
```

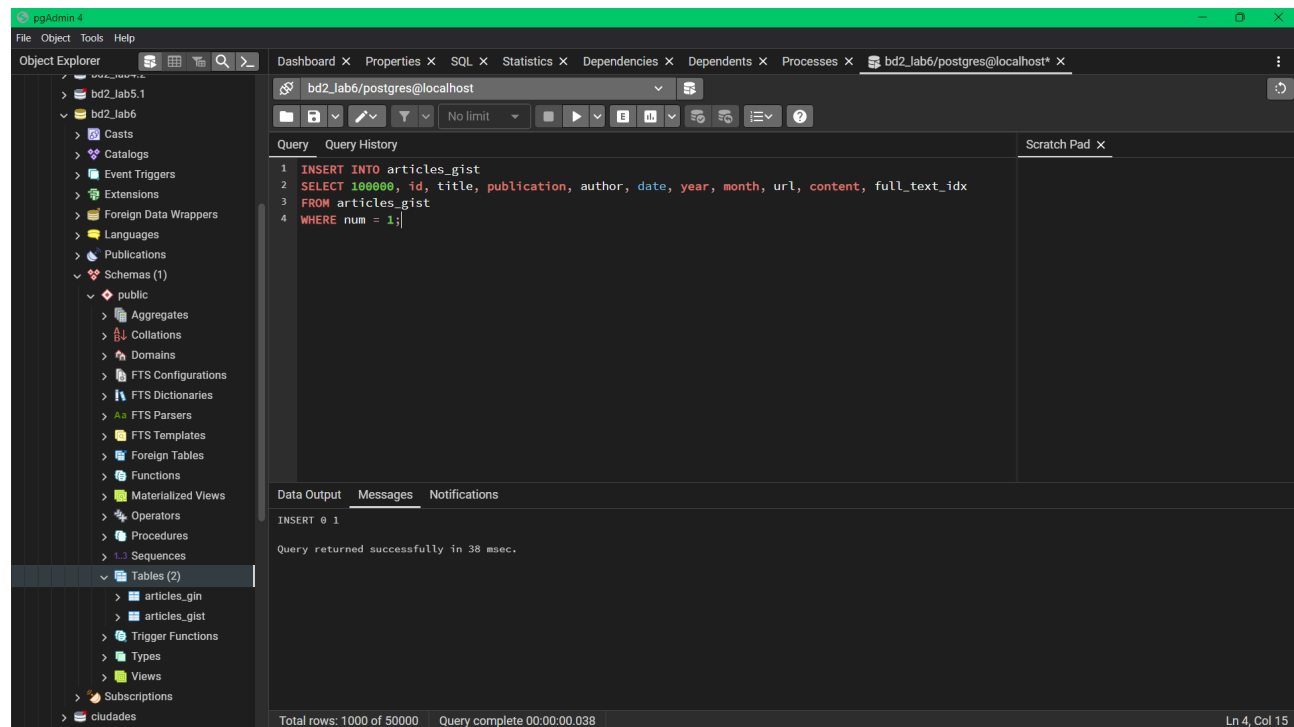
```
INSERT INTO articles_gist
SELECT 100000, id, title, publication, author, date, year, month, url, content, full_text_idx
FROM articles_gist
WHERE num = 1;
```

Tiempo de inserción:

Índice gin:



Índice gist:



La inserción con el índice gin demoró 40ms, y con el índice gist demoró 38ms, prácticamente el mismo tiempo.

Ahora compararemos el tiempo de búsqueda usando las siguientes consultas:

```
EXPLAIN ANALYZE
SELECT * FROM articles_gin
WHERE full_text_idx @@ plainto_tsquery('english', 'trump');
```

```
EXPLAIN ANALYZE
SELECT * FROM articles_gist
WHERE full_text_idx @@ plainto_tsquery('english', 'trump');
```

Se tuvo que deshabilitar el sequential scan para forzar el uso del índice.

Tiempo de búsqueda:

Índice gin:

pgAdmin 4

Object Explorer

- bd2_jab5.1
- bd2_jab6
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (2)
 - articles_gin
 - articles_gist
 - Trigger Functions
 - Types
 - Views
 - Subscriptions
 - ciudades

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Processes X bd2_jab6/postgres@localhost X

bd2_jab6/postgres@localhost

Query Query History

```

1 VACUUM FULL
2
3 EXPLAIN ANALYZE
4 SELECT * FROM articles_gin
5 WHERE full_text_idx @@ plainto_tsquery('english', 'trump');
6
7 EXPLAIN ANALYZE
8 SELECT * FROM articles_gist
9 WHERE full_text_idx @@ plainto_tsquery('english', 'trump');
10

```

Data Output Messages Notifications

QUERY PLAN

1	Bitmap Heap Scan on articles_gin (cost=150.29..5542.66 rows=19870 width=794) (actual time=2.194..28.690 rows=19907 loops=1)
2	Recheck Cond: (full_text_idx @@ "trump":tsquery)
3	Heap Blocks: exact=4700
4	Bitmap Index Scan on gin_articles_idx (cost=0.00..145.32 rows=19870 width=0) (actual time=1.673..1.673 rows=19907 loops=1)
5	Index Cond: (full_text_idx @@ "trump":tsquery)
6	Planning Time: 5.866 ms
7	Execution Time: 29.169 ms

Total rows: 7 of 7 Query complete 00:00:00.066 Ln 3, Col 1

Índice gist:

pgAdmin 4

Object Explorer

- bd2_jab5.1
- bd2_jab6
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (2)
 - articles_gin
 - articles_gist
 - Trigger Functions
 - Types
 - Views
 - Subscriptions
 - ciudades

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Processes X bd2_jab6/postgres@localhost X

bd2_jab6/postgres@localhost

Query Query History

```

1 VACUUM FULL
2
3 EXPLAIN ANALYZE
4 SELECT * FROM articles_gin
5 WHERE full_text_idx @@ plainto_tsquery('english', 'trump');
6
7 EXPLAIN ANALYZE
8 SELECT * FROM articles_gist
9 WHERE full_text_idx @@ plainto_tsquery('english', 'trump');
10

```

Data Output Messages Notifications

QUERY PLAN

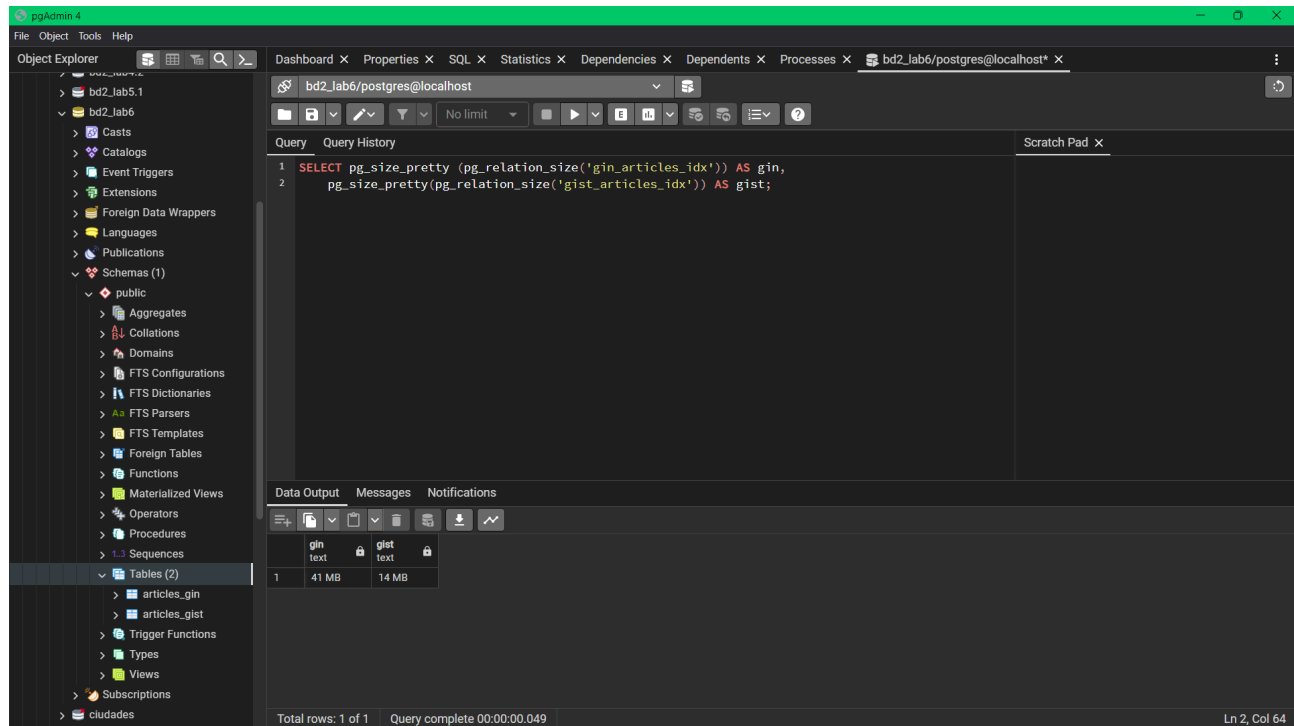
1	Recheck Cond: (full_text_idx @@ "trump":tsquery)
2	Rows Removed by Index Recheck: 2280
3	Heap Blocks: exact=4842
4	Bitmap Index Scan on gist_articles_idx (cost=0.00..2918.18 rows=19987 width=0) (actual time=13.845..13.845 rows=22187 loops=1)
5	Index Cond: (full_text_idx @@ "trump":tsquery)
6	Planning Time: 0.326 ms
7	Execution Time: 234.430 ms

Total rows: 8 of 8 Query complete 00:00:00.260 Ln 7, Col 1

La búsqueda con el índice gin demoró 29.169 ms, mientras que con el índice gist demoró 234.430 ms. Se puede ver cómo el índice gin logró una consulta mucho más rápida.

Ahora compararemos el tamaño en disco de ambos índices. Ejecutamos la siguiente consulta:


```
SELECT pg_size_pretty (pg_relation_size('gin_articles_idx')) AS gin,
pg_size_pretty(pg_relation_size('gist_articles_idx')) AS gist;
```



El índice gin ocupa 41MB, y el índice gist ocupa 14MB. Con esto podemos ver que el índice gin pesa mucho más que el índice gist.

Resumiendo todos los resultados en una tabla, tenemos lo siguiente:

	GIN	GiST
Tiempo de creación (s)	7.156	1.968
Tiempo de inserción (ms)	40	38
Tiempo de búsqueda (ms)	29.169	234.430
Tamaño en disco (MB)	41	14

El tiempo de creación no significa mucho, ya que solo se toma en cuenta una única vez, al crear el índice. Igual cabe mencionar que el índice gin demoró mucho más tiempo en crearse. El tiempo de inserción es prácticamente idéntico en ambos índices. Sin embargo, el tiempo de búsqueda nos da a conocer una diferencia importante. El índice gist demoró aproximadamente 8 veces más en dar una respuesta, por lo que el gin es mucho más eficiente. A cambio de esta mejora en el tiempo de búsqueda, podemos ver cómo el índice gin ocupa mucho más espacio en disco, por lo que esa mejora en rapidez tiene un costo.

Para concluir, podemos decir que el índice gin ofrece un mejor rendimiento que el índice gist, a cambio de un aumento en el espacio que ocupa en disco.