



**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ
ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра системного програмування та спеціалізованих комп'ютерних систем**

**Лабораторна робота №3
з дисципліни Базы даних і засоби управління
на тему: «Засоби оптимізації роботи СУБД PostgreSQL»**

Виконав:
студент III курсу
групи КВ-92
Михайлов О.Г.
Перевірив:
Петрашенко А.В.

Метою роботи є здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

Завдання роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.
4. Навести приклади та проаналізувати рівні ізоляції транзакцій у PostgreSQL.

<i>№ варіанта</i>	<i>Види індексів</i>	<i>Умови для тригера</i>
<i>14</i>	<i>BTree, BRIN</i>	<i>before delete, update</i>

URL репозиторію з вихідним кодом:

https://github.com/AlexMykhalov/DataBase_KV-92.git

Завдання №1

Модель «сутність-зв'язок»:

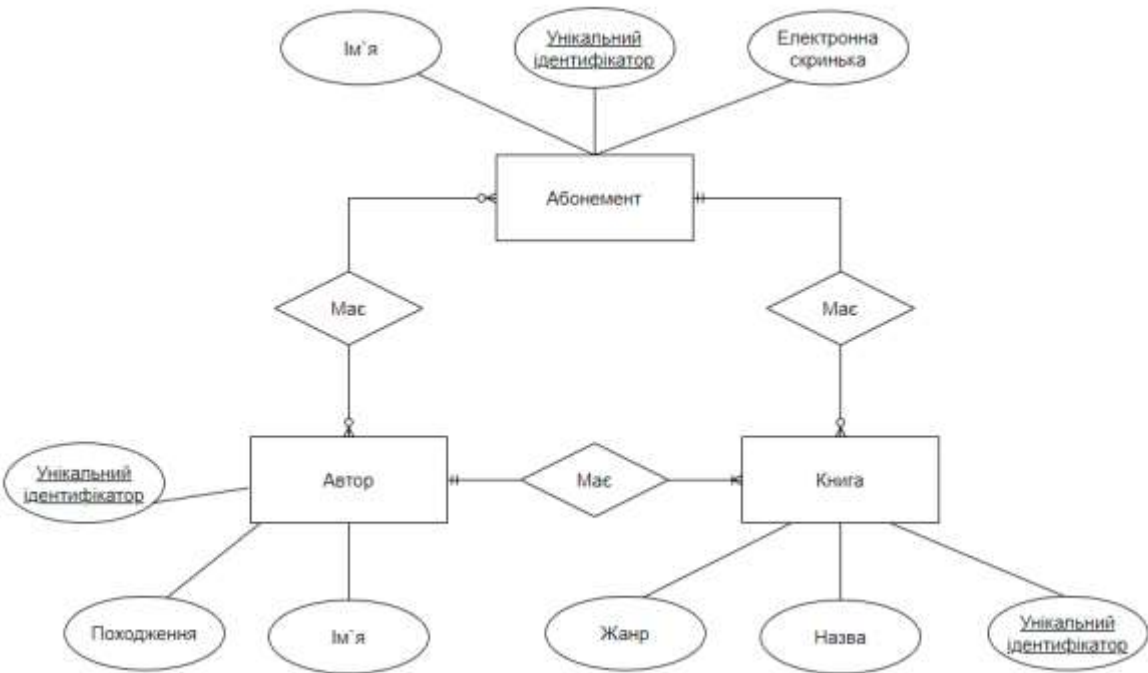
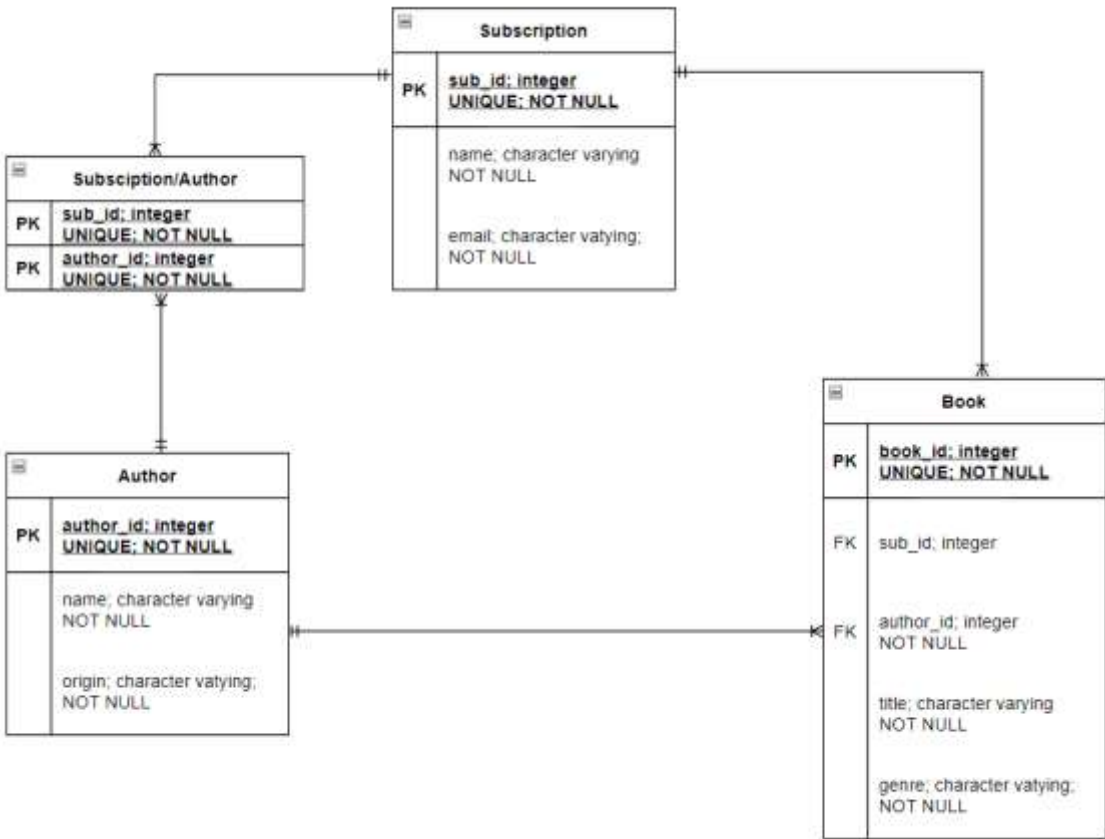


Рис1. ER-діаграма за нотацією Чена

Схеми бази даних:



Класи ORM, що відповідають таблицям бази даних

```
class Author(Base):
    __tablename__ = 'Author'
    author_id = Column(Integer, primary_key=True)
    name = Column(String)
    origin = Column(String)

    authors = relationship('Book')

    def __init__(self, author_id: int, name: str, origin: str):
        self.author_id = author_id
        self.name = name
        self.origin = origin

class Book(Base):
    __tablename__ = 'Book'
    book_id = Column(Integer, primary_key=True)
    sub_id = Column(Integer, ForeignKey('Author.author_id'))
    author_id = Column(Integer, ForeignKey('Subscription.sub_id'))
    title = Column(String)
    genre = Column(String)

    def __init__(self, book_id: int, sub_id: int, author_id: int, title: str,
genre: str):
        self.book_id = book_id
        self.sub_id = sub_id
        self.author_id = author_id
        self.title = title
        self.genre = genre

class Subscription(Base):
    __tablename__ = 'Subscription'
    sub_id = Column(Integer, primary_key=True)
    name = Column(String)
    email = Column(String)

    subscriptions = relationship('Book')

    def __init__(self, sub_id: int, name: str, email: str):
        self.sub_id = sub_id
        self.name = name
        self.email = email
```

Результат реалізації вставки, видалення та редагування

Вставка:

```
-----
| 1 to insert          |
| 2 to insert random   |
| 3 to update          |
| 4 to delete          |
| 5 to select table    |
| 6 to search          |
| 7 to end             |
|-----|
Input: 1
-----
| Select a table:      |
| 1 = Author          |
| 2 = Book             |
| 3 = Subscription    |
|-----|
Input: 1
Input new author_id: 555
Input new author name: O. Henry
Input new origin: USA

[Data was successfully inserted]
```

Before:

21	25	Molly Gill	Madagascar
22	26	Kenneth Chapman	Paraguay
23	27	Christopher Winters	Saint Kitts and Nevis
24	4212	Eric Arthur Blair	India
25	46546	Joanne Rowling	England
26	54454	212	121212
27	213689	Agatha Mary Clarissa Miller	England

After:

22	26	Kenneth Chapman	Paraguay
23	27	Christopher Winters	Saint Kitts and Nevis
24	555	O. Henry	USA
25	4212	Eric Arthur Blair	India
26	46546	Joanne Rowling	England
27	54454	212	121212
28	213689	Agatha Mary Clarissa Miller	England

Вилучення:

```
-----
| 1 to insert      |
| 2 to insert random |
| 3 to update      |
| 4 to delete      |
| 5 to select table |
| 6 to search      |
| 7 to end         |
|-----|
Input: 4
-----
| Select a table:  |
| 1 = Author      |
| 2 = Book         |
| 3 = Subscription |
|-----|
Input: 2
Num of id: 100

[Data was successfully deleted]
```

Before:

3	4	2	4212	1984	dystopian	
4	5	1	46546	NRGHZ	WPPVN	
5	6	3	19	HCNJP	YCBWF	
6	7	5	4	OIUXD	XSRHV	
7	100	1	46546	The Ickabog	Fairy tale	

After:


3	4	2	4212	1984	dystopian	
4	5	1	46546	NRGHZ	WPPVN	
5	6	3	19	HCNJP	YCBWF	
6	7	5	4	OIUXD	XSRHV	

Редагування:





```
-----
| 1 to insert      |
| 2 to insert random |
| 3 to update      |
| 4 to delete      |
| 5 to select table |
| 6 to search      |
| 7 to end         |
|-----|
Input: 3
-----
| Select a table:  |
| 1 = Author      |
| 2 = Book         |
| 3 = Subscription |
|-----|
Input: 3
Num of id: 5
1 = to upd name
2 = to upd email
Input: 1
New name: Ivan

[Data was successfully updated]
```

Before:

		sub_id [PK] integer 	name character varying 	email character varying 
	1	1	Alex	alex@gmail.com
	2	2	Petro	peter@gmail.com
	3	3	Anton	anton@gmail.com
	4	4	DWVPJ	SOJEZ@email.com
	5	5	FEJJN	XEUHV@email.com

After:

		sub_id [PK] integer 	name character varying 	email character varying 
	1	1	Alex	alex@gmail.com
	2	2	Petro	peter@gmail.com
	3	3	Anton	anton@gmail.com
	4	4	DWVPJ	SOJEZ@email.com
	5	5	Ivan	XEUHV@email.com

Результат реалізації пошуку та генерації даних

Пошук:

```
-----  
| 1 to insert      |  
| 2 to insert random |  
| 3 to update      |  
| 4 to delete      |  
| 5 to select table |  
| 6 to search      |  
| 7 to end         |  
-----
```

Input: 6

```
-----  
| Select a table:  |  
| 1 = Author       |  
| 2 = Book         |  
| 3 = Subscription |  
-----
```

Input: 3

1 = to search sub_id

2 = to search name

3 = to search email

Input: 1

Input start: 1

Input finish: 3

```
sub_id = 1  
name = Alex  
email = alex@gmail.com
```

```
sub_id = 2  
name = Petro  
email = peter@gmail.com
```

```
sub_id = 3  
name = Anton  
email = anton@gmail.com
```

Search duration = 0.005983114242553711

Генерація даних:

```
-----
| 1 to insert |
| 2 to insert random |
| 3 to update |
| 4 to delete |
| 5 to select table |
| 6 to search |
| 7 to end |
-----

Input: 2

-----
| Select a table: |
| 1 = Author |
| 2 = Book |
| 3 = Subscription |
-----

Input: 1
Input amount of data to generate: 4

[Random Data was successfully inserted]
```

Before:

14	18	FHRHX	FFVDD
15	19	KIDIV	[OZJW
16	20	PSUEH	COIUX
17	21	RMLVV	YSDSQ
18	22	EDHFG	TONDO
19	23	Cheryl Barton	Samoa
20	24	Christopher Harvey	Norfolk Island
21	25	Molly Gill	Madagascar
22	26	Kenneth Chapman	Paraguay
23	27	Christopher Winters	Saint Kitts and Nevis
24	4212	Eric Arthur Blair	India
25	46546	Joanne Rowling	England
26	54454	212	121212
27	213689	Agatha Mary Clarissa Miller	England

After:

22	26	Kenneth Chapman	Paraguay
23	27	Christopher Winters	Saint Kitts and Nevis
24	28	Nicholas Mcdaniel	Equatorial Guinea
25	29	April Mcdonald	Cote d'Ivoire
26	30	Darren Thomas	Netherlands Antilles
27	31	Phillip Nichols	Kenya
28	4212	Eric Arthur Blair	India
29	46546	Joanne Rowling	England

Завдання №2

BTree

Запити для тестування:

explain analyze select * from "Author" where origin = 'Mali' ;

explain analyze select * from "Author" where origin = 'Mali'
and origin = 'Chile' and origin = 'Panama';

explain analyze select * from "Author" where origin != 'Mali'
and origin != 'Chile' and origin != 'Panama' and origin != 'Ukraine';

explain analyze select * from "Film" where id_film < 10000 ORDER BY origin;

Результати тестування:

DROP INDEX if exists or_index;

```
Library=# DROP INDEX if exists or_index;
DROP INDEX
Time: 3,838 ms
Library=# explain analyze select * from "Author" where origin = 'Mali' ;
               QUERY PLAN
-----
Seq Scan on "Author" (cost=0.00..2006.18 rows=402 width=29) (actual time=0.018..9.116 rows=428 loops=1)
  Filter: ((origin)::text = 'Mali'::text)
  Rows Removed by Filter: 99586
  Planning Time: 1.456 ms
  Execution Time: 9.136 ms
(5 rows)

Time: 11,368 ms
Library=# explain analyze select * from "Author" where origin = 'Mali' and origin = 'Chile' and origin = 'Panama';
               QUERY PLAN
-----
Result (cost=0.00..2006.18 rows=1 width=29) (actual time=0.000..0.001 rows=0 loops=1)
  One-Time Filter: (false AND false)
  -> Seq Scan on "Author" (cost=0.00..2006.18 rows=1 width=29) (never executed)
    Filter: ((origin)::text = 'Mali'::text)
  Planning Time: 0.095 ms
  Execution Time: 0.009 ms
(6 rows)

Time: 0,425 ms
Library=# explain analyze select * from "Author" where origin != 'Mali' and origin != 'Chile' and origin != 'Panama' and origin != 'Ukraine';
               QUERY PLAN
-----
Seq Scan on "Author" (cost=0.00..2736.28 rows=98416 width=29) (actual time=0.008..14.160 rows=98333 loops=1)
  Filter: (((origin)::text <> 'Mali'::text) AND ((origin)::text <> 'Chile'::text) AND ((origin)::text <> 'Panama'::text) AND ((origin)::text <> 'Ukraine'::text))
  Rows Removed by Filter: 1061
  Planning Time: 0.051 ms
  Execution Time: 15.902 ms
(3 rows)

Library=# explain analyze select * from "Author" where author_id < 10000 ORDER BY origin;
               QUERY PLAN
-----
Sort (cost=1112.78..1139.64 rows=10745 width=29) (actual time=15.168..15.489 rows=9999 loops=1)
  Sort Key: origin
  Sort Method: quicksort  Memory: 1187kB
  -> Index Scan using "Author_pkey" on "Author" (cost=0.29..393.33 rows=10745 width=29) (actual time=0.052..1.194 rows=9999 loops=1)
    Index Cond: (author_id < 10000)
  Planning Time: 0.117 ms
  Execution Time: 15.867 ms
(7 rows)

Time: 16,519 ms
Library=#
```

CREATE INDEX or index ON “Author”(origin);

```
library=# CREATE INDEX or_index ON "Author"(origin);
CREATE INDEX
Time: 268,832 ms
library=# explain analyze select * from "Author" where origin = 'Mali' ;
QUERY PLAN
-----
Bitmap Heap Scan on "Author" (cost=7.41..665.70 rows=482 width=29) (actual time=0.101..0.296 rows=428 loops=1)
  Recheck Cond: (((origin)::text = 'Mali'::text))
  Heap Blocks: exact=325
  -> Bitmap Index Scan on or_index (cost=0.00..7.31 rows=482 width=0) (actual time=0.075..0.075 rows=428 loops=1)
    Index Cond: ((origin)::text = 'Mali'::text)
Planning Time: 1.319 ms
Execution Time: 0.321 ms
(7 rows)

Time: 1,959 ms
library=# explain analyze select * from "Author" where origin = 'Mali' and origin = 'Chile' and origin = 'Panama';
QUERY PLAN
-----
Result (cost=7.31..665.60 rows=1 width=29) (actual time=0.000..0.001 rows=0 loops=1)
  One-Time Filter: (false AND false)
  -> Bitmap Heap Scan on "Author" (cost=7.31..665.60 rows=1 width=29) (never executed)
    Recheck Cond: (((origin)::text = 'Mali'::text))
    -> Bitmap Index Scan on or_index (cost=0.00..7.31 rows=482 width=0) (never executed)
      Index Cond: ((origin)::text = 'Mali'::text)
Planning Time: 0.001 ms
Execution Time: 0.019 ms
(8 rows)

Time: 0,569 ms
library=# explain analyze select * from "Author" where origin != 'Mali' and origin != 'Chile' and origin != 'Panama' and origin != 'Ukraine';
QUERY PLAN
-----
Seq Scan on "Author" (cost=0.00..12756.28 rows=98416 width=29) (actual time=0.012..17.288 rows=98351 loops=1)
  Filter: (((origin)::text <> 'Mali'::text) AND ((origin)::text <> 'Chile'::text) AND ((origin)::text <> 'Panama'::text) AND ((origin)::text <> 'Ukraine'::text))
  Rows Removed by Filter: 1661
Planning Time: 0.144 ms
Execution Time: 19.048 ms
(5 rows)

Time: 19,647 ms
library=# explain analyze select * from "Author" where author_id < 10000 ORDER BY origin;
QUERY PLAN
-----
Sort (cost=1112.78..1139.64 rows=10745 width=29) (actual time=15.194..15.501 rows=9999 loops=1)
  Sort Key: origin
  Sort Method: quicksort Memory: 1187kB
  -> Index Scan using "Author_pkey" on "Author" (cost=0.29..393.33 rows=10745 width=29) (actual time=0.043..1.141 rows=9999 loops=1)
    Index Cond: (author_id < 10000)
Planning Time: 0.007 ms
Execution Time: 15.872 ms
(7 rows)

Time: 16,308 ms
```

Отже з отриманих результатів бачимо що, пошук з індексацією відбувається набагато швидше для роботи з невеликою кількістю даних, ніж без індексації. ВТрее має свою особливість, а саме, він показує свою ефективність тоді коли працює з невеликою кількістю даних. При сортуванні даних з індексацією бачимо, що вона відбулась дещо швидше ніж без індексації, це пов'язано також з нормальною кількістю даних для індексу ВТрее.

BRIN

Запити для тестування:

explain analyze select * from "Author" where origin = 'Mali' ;

explain analyze select count(name) from "Author" where origin != 'Mali';

explain analyze select count(name) from "Author" where origin = 'Mali' and origin = 'Chile' and origin = 'Panama';

explain analyze select count(name) from "Author" where origin != 'Mali' and origin != 'Chile' and origin != 'Panama' and origin != 'Ukraine';

Результати тестування:

CREATE INDEX or_brin_index on "Author" using brin (origin) with(pages_per_range=128); with(pages_per_range=128);

```
Library=# CREATE INDEX or_brin_index on "Author" using brin (origin) with(pages_per_range=128);
CREATE INDEX
Time: 90,767 ms
Library=# explain analyze select * from "Author" where origin = 'Mali' ;
               QUERY PLAN
-----
Seq Scan on "Author"  (cost=0.00..2006.18 rows=402 width=29) (actual time=0.018..8.108 rows=428 loops=1)
  Filter: ((origin)::text = 'Mali'::text)
  Rows Removed by Filter: 99586
Planning Time: 1.847 ms
Execution Time: 8.129 ms
(5 rows)

Time: 10,389 ms
Library=# explain analyze select count(name) from "Author" where origin != 'Mali';
               QUERY PLAN
-----
Aggregate  (cost=2255.21..2255.22 rows=1 width=0) (actual time=11.850..11.851 rows=1 loops=1)
-> Seq Scan on "Author"  (cost=0.00..2006.18 rows=99612 width=14) (actual time=0.008..8.493 rows=99586 loops=1)
  Filter: ((origin)::text <> 'Mali'::text)
  Rows Removed by Filter: 428
Planning Time: 0.885 ms
Execution Time: 11.871 ms
(6 rows)

Time: 12,411 ms
Library=# explain analyze select count(name) from "Author" where origin = 'Mali' and origin = 'Chile' and origin = 'Panama';
               QUERY PLAN
-----
Aggregate  (cost=2006.18..2006.19 rows=1 width=0) (actual time=0.002..0.002 rows=1 loops=1)
-> Result  (cost=0.00..2006.18 rows=1 width=14) (actual time=0.000..0.000 rows=0 loops=1)
  One-Time Filter: (False AND false)
  -> Seq Scan on "Author"  (cost=0.00..2006.18 rows=1 width=14) (never executed)
    Filter: ((origin)::text = 'Mali'::text)
Planning Time: 0.096 ms
Execution Time: 0.021 ms
(7 rows)

Time: 9,483 ms
Library=# explain analyze select count(name) from "Author" where origin != 'Mali' and origin != 'Chile' and origin != 'Panama' and origin != 'Ukraine';
               QUERY PLAN
-----
Aggregate  (cost=3002.32..3002.33 rows=1 width=0) (actual time=17.406..17.407 rows=1 loops=1)
-> Seq Scan on "Author"  (cost=0.00..2756.28 rows=98416 width=14) (actual time=0.011..14.073 rows=98353 loops=1)
  Filter: (((origin)::text <> 'Mali'::text) AND ((origin)::text <> 'Chile'::text) AND ((origin)::text <> 'Panama'::text) AND ((origin)::text <> 'Ukraine'::text))
  Rows Removed by Filter: 1661
Planning Time: 0.872 ms
Execution Time: 17.426 ms
(6 rows)

Time: 17,878 ms
```

DROP INDEX if exists or_brin_index;

```
Library=# DROP INDEX if exists or_brin_index;
DROP INDEX
Time: 3,360 ms
Library=# explain analyze select * from "Author" where origin = 'Mali' ;
               QUERY PLAN
-----
Seq Scan on "Author"  (cost=0.00..2006.18 rows=402 width=29) (actual time=0.012..6.881 rows=428 loops=1)
  Filter: ((origin)::text = 'Mali')::text)
  Rows Removed by Filter: 99586
  Planning Time: 0.943 ms
  Execution Time: 6.906 ms
(5 rows)

Time: 8,200 ms
Library=# explain analyze select count(name) from "Author" where origin != 'Mali';
               QUERY PLAN
-----
Aggregate  (cost=2255.21..2255.22 rows=1 width=8) (actual time=12.329..12.330 rows=1 loops=1)
-> Seq Scan on "Author"  (cost=0.00..2006.18 rows=99612 width=14) (actual time=0.013..8.823 rows=99586 loops=1)
  Filter: ((origin)::text <> 'Mali')::text)
  Rows Removed by Filter: 428
  Planning Time: 0.064 ms
  Execution Time: 12.359 ms
(6 rows)

Time: 12,779 ms
Library=# explain analyze select count(name) from "Author" where origin = 'Mali' and origin = 'Chile' and origin = 'Panama';
               QUERY PLAN
-----
Aggregate  (cost=2006.18..2006.19 rows=1 width=8) (actual time=0.003..0.003 rows=1 loops=1)
-> Result  (cost=0.00..2006.18 rows=1 width=14) (actual time=0.000..0.000 rows=0 loops=1)
  One-Time Filter: (false AND false)
  -> Seq Scan on "Author"  (cost=0.00..2006.18 rows=1 width=14) (never executed)
    Filter: ((origin)::text = 'Mali')::text)
  Planning Time: 0.087 ms
  Execution Time: 0.020 ms
(7 rows)

Time: 0,486 ms
Library=# explain analyze select count(name) from "Author" where origin != 'Mali' and origin != 'Chile' and origin != 'Panama' and origin != 'Ukraine';
               QUERY PLAN
-----
Aggregate  (cost=3002.12..3002.13 rows=1 width=8) (actual time=17.727..17.728 rows=1 loops=1)
-> Seq Scan on "Author"  (cost=0.00..2750.28 rows=98426 width=14) (actual time=0.012..14.223 rows=98351 loops=1)
  Filter: (((origin)::text <> 'Mali')::text) AND ((origin)::text <> 'Chile')::text) AND ((origin)::text <> 'Panama')::text) AND ((origin)::text <> 'Ukraine')::text))
  Rows Removed by Filter: 1861
  Planning Time: 0.132 ms
  Execution Time: 17.763 ms
(6 rows)

Time: 18,279 ms
```

BRIN показує не сильно швидші результати ніж без індексування, в деяких випадках воно швидше, а в деяких повільніше, ніж без індексування. Але це пов'язано з особливістю даного індексу а саме не в швидкодія знаходження потрібних рядків, а уникнення перегляду зарання непотрібних даних.

Завдання №3

Створення та заповнення таблиць баз даних для роботи з тригером:

```
CREATE TABLE "book"(  
  "book_id" serial PRIMARY KEY,  
  "book_title" text,  
  "book_pages" int  
);
```

```
CREATE TABLE "book_log"(  
  "id" serial PRIMARY KEY,  
  "book_log_id" int,  
  "book_log_name" text  
);
```

```
INSERT INTO "book"("book_title", "book_pages")  
VALUES ('book1', '100'), ('book2', '200'), ('book3', '300'), ('book4', '400'),  
('book5', '500'), ('book6', '600'), ('book7', '700'), ('book8', '800'),  
('book9', '900'), ('book9', '1000');
```

Створення тригера:

```
CREATE OR REPLACE FUNCTION before_delete_update_func() RETURNS TRIGGER as  
$trigger$  
DECLARE
```

```
BEGIN
```

```
  IF old."book_pages" <= 500 THEN  
    RAISE NOTICE 'book_pages <= 500';  
    INSERT INTO "book_log"("book_log_id", "book_log_name") VALUES  
    (old."book_id", old."book_title" || '_short');  
    RETURN OLD;
```

```
  ELSE  
    RAISE NOTICE 'book_pages >= 500';  
    INSERT INTO "book_log"("book_log_id", "book_log_name") VALUES  
    (old."book_id", old."book_title" || '_long');  
    RETURN OLD;
```

```
  END IF;
```

```
END;
```

```
$trigger$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER "before_delete_update_trigger"  
BEFORE DELETE OR UPDATE ON "book"  
FOR EACH ROW  
EXECUTE procedure before_delete_update_func();
```

Запити для перевірки роботи тригера:

```
DELETE FROM "book" where "book_id" = 4;
```

```
UPDATE "book" SET "book_title" = "book_title" WHERE "book_pages" = 800;
```

Результати роботи тригера

Вміст таблиць до початку роботи з ними:

Таблиця book:

	book_id [PK] integer	book_title text	book_pages integer
1	1	book1	100
2	2	book2	200
3	3	book3	300
4	4	book4	400
5	5	book5	500
6	6	book6	600
7	7	book7	700
8	8	book8	800
9	9	book9	900
10	10	book9	1000

Таблиця book_log:

	id [PK] integer	book_log_id integer	book_log_name text

Вміст таблиць після виконання запитів

DELETE FROM "book" where "book_id" = 4;

Таблиця book:

	book_id [PK] integer	book_title text	book_pages integer
1	1	book1	100
2	2	book2	200
3	3	book3	300
4	5	book5	500
5	6	book6	600
6	7	book7	700
7	8	book8	800
8	9	book9	900
9	10	book9	1000

Таблиця book_log:

	id [PK] integer	book_log_id integer	book_log_name text
1	1	4	book4_short

Видно, що запит був виконаний правильно та тригерна функція спрацювала корктно тому, що в таблиці book був видалений потрібний рядок, а в таблицю book_log внесений новий запис, а саме рядок який ми видалили з дописаним укінці “_short”.

UPDATE "book" SET "book_title" = "book_title" WHERE "book_pages" = 800;

Таблиця book:

	book_id [PK] integer	book_title text	book_pages integer
1	1	book1	100
2	2	book2	200
3	3	book3	300
4	4	book4	400
5	5	book5	500
6	6	book6	600
7	7	book7	700
8	8	book8	800
9	9	book9	900
10	10	book9	1000

Таблиця book_log:

	id [PK] integer	book_log_id integer	book_log_name text
1	1	8	book8_long

Запит, виконує оновлення назви книги на те саме, тому в таблиці book нічого не зміниться, а в таблицю book_log був внесений новий запис оновленого рядка з попередньої таблиці з приставкою вкінці “_long”, що підтверджує правильність виконання запиту та роботи тригерної функції