



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  
**«Бази даних і засоби управління»**

**Тема: «Засоби оптимізації роботи СУБД  
PostgreSQL»**

Виконав: студент III курсу

ФПМ групи КВ-94

Зінчук М. С.

Перевірів: Петрашенко А.В.

Київ 2021

### Завдання роботи:

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.

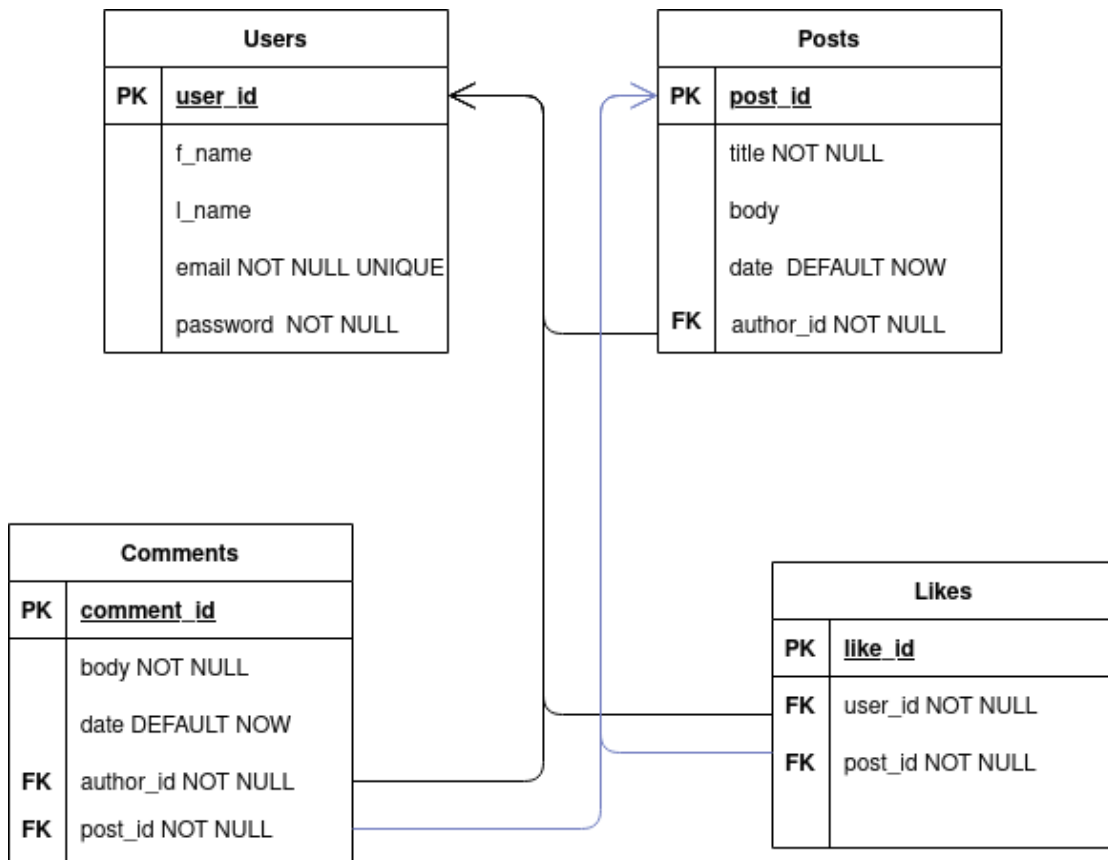
Git: [Репозиторій на гітхабі](#)

### Варіант 8

8	<i>BTree, GIN</i>	<i>after insert, update</i>
---	-------------------	-----------------------------

### Завдання 1

Схема бд:



## Insert:

### Table: posts

```
PRESS: 1 to add... 2 to update... 3 to delete... 4 to specific_select... 5 to show_table...
1
Choose table:
press 1 - users... press 2 - posts... press 3 - likes... press 4 - comments
2
How much rows do you wanna add?
1
Enter values following this sequence: title, body, author_id
title: Good memories
body: sfdkfsdafksdsdafsd
author_id: 1
Successfully inserted
Do you want to continue? Press Y if yes, N if no:
```

post_id	title	body	date	author_id
2	New PC	So last week i bought a new PC...	2021-09-11 14:17:45.551787+03:00	2
1164019	Good memories	sfdkfsdafksdsdafsd	2021-12-23 13:50:41.004283+02:00	1
4	My last job	fsdfsdksdkfsdkfsdksdkfsd	2021-09-11 14:18:28.289647+03:00	3
5	Msdkfsdafsd	sfdksdafsd	2021-10-31 16:16:09.707603+02:00	1
6	fsdf	fsdffsd	2021-10-31 21:49:22.683802+02:00	3
8	f499491fa6658cfd0b239dbb39b16f00	699ae7e2693a253ee271ac1f8de00f5e	2021-10-31 23:24:41.961766+02:00	3
9	1a044eda4a44bc7accaedecd4a9990f1	53a41b941e30ef2cae9b8457c1ca9371	2021-10-31 23:24:41.961766+02:00	3
7	Turkey	very beaut...	2021-10-31 23:24:41.961766+02:00	5
1	aibaiba	aibaibabi	2021-09-11 14:16:42.811920+03:00	3

### Error handling:

```
Choose table:
press 1 - users... press 2 - posts... press 3 - likes... press 4 - comments
3
How much rows do you wanna add?
1
Enter values following this sequence: title, body, author_id
title: Top meetup
body: fsdfsd
author_id: 100
Failed inserting record into table (psycopg2.errors.ForeignKeyViolation) insert or update on table "posts" violates foreign key constraint: Key (author_id)=(100) is not present in table "users".

[SQL: INSERT INTO posts (author_id, title, body, date) VALUES (%(author_id)s, %(title)s, %(body)s, %(date)s) RETURNING posts.post_id]
[parameters: {'author_id': '100', 'title': 'Top meetup', 'body': 'fsdfsd', 'date': datetime.datetime(2021, 12, 23, 13, 52, 8, 957609)}]
(Background on this error at: https://sqlalche.me/e/14/gkpi)
Do you want to continue? Press Y if yes, N if no:
```

### Table: likes

```

PRESS: 1 to add... 2 to update... 3 to delete... 4 to specific_select... 5 to show_table...
1
Choose table:
press 1 - users... press 2 - posts... press 3 - likes... press 4 - comments
3
How much rows do you wanna add?
1
Enter values following this sequence: user_id, post_id
user_id: 5
post_id: 4
Successfully inserted

```

like_id	user_id	post_id
101020	3	1
101025	5	4

## Error handling:

```

Enter values following this sequence: user_id, post_id
user_id: 7
post_id: 1
Failed inserting record into table (psycopg2.errors.ForeignKeyViolation) insert or update on table "likes" violates foreign key constraint: Key (user_id)=(7) is not present in table "users".
DETAIL: Key (user_id)=(7) is not present in table "users".

[SQL: INSERT INTO likes (user_id, post_id) VALUES (%(user_id)s, %(post_id)s) RETURNING likes.like_id]
[parameters: {'user_id': '7', 'post_id': '1'}]
(Background on this error at: https://sqlalche.me/e/14/gkpg)
Do you want to continue? Press Y if yes, N if no: |

```

## Table: Comments

```

How much rows do you wanna add?
1
Enter values following this sequence: body, author_id, post_id
body: Nice info
user_id: 1
post_id: 4
Successfully inserted

```

comment_id	body	date	author_id	post_id
100020	Nice info	2021-12-23 14:16:42.763118+02:00	1	4

## Error handling:

```
Enter values following this sequence: body, author_id, post_id
body: fsdfsd
user_id: 14
post_id: 1
Failed inserting record into table (psycopg2.errors.ForeignKeyViolation) insert or update on table "comments" violates foreign key co
DETAIL: Key (author_id)=(14) is not present in table "users".

[SQL: INSERT INTO comments (author_id, post_id, body, date) VALUES (%(author_id)s, %(post_id)s, %(body)s, %(date)s) RETURNING comment
[parameters: {'author_id': '14', 'post_id': '1', 'body': 'fsdfsd', 'date': datetime.datetime(2021, 12, 23, 14, 17, 24, 22785)}]
(Background on this error at: https://sqlalche.me/e/14/gkpi)
Do you want to continue? Press Y if yes, N if no: |
```

## Table: Users

```
Choose table:
press 1 - users...   press 2 - posts...   press 3 - likes...   press 4 - comments
1
How much rows do you wanna add?
1
Enter values following this sequence: f_name, l_name, email, password
f_name: Mokko
l_name: Darrinson
email: dskfsdfs@gmail.com
password: dsfksdfsd
Successfully inserted
```

8	8	Vasiluk	igor@gmail.coom	flsdfkwe
2	2	Danes	johndanes@gmail.com	qwerty
3	3	Banes	darek@gmail.com	zxcvbn
452848	452848	fsdafsdafsd	safdsfas	sdfasdfsdfsd
6	6	fsdfsd	123214sa	fsddfsd
5	5	аыбацук	zenya@gmail.com2	аыыаыы
9	9	SHKCWL	YKAJJL	HNPSKL
452850	452850	Darrinson	dskfsdfs@gmail.com	dsfksdfsd
1	1	fsdfsd	fsdfsd	sdfsd

## Update:

### Table: Posts

post_id	title	body	date	author_id
2	New PC	So last week i bought a new PC....	2021-09-11 14:17:45.551787+03:00	2
1164019	Good memories	sdfksdafksdsdafsd	2021-12-23 13:50:41.004283+02:00	1
4	My last job	fsdfsdfsdksdfdsfksdfsd	2021-09-11 14:18:28.289647+03:00	3
5	Msdksdafsd	sfsadfsd	2021-10-31 16:16:09.707603+02:00	1
6	fsdf	fsdffd	2021-10-31 21:49:22.683802+02:00	3
8	f499491fa6658cfd0b239dbb39b16f00	699ae7e2693a253ee271ac1f8de00f5e	2021-10-31 23:24:41.961766+02:00	3
9	1a044eda4a44bc7accaedecd4a9990f1	53a41b941e30ef2cae9b8457c1ca9371	2021-10-31 23:24:41.961766+02:00	3
1	fsdf	sdfsdfsd	2021-09-11 14:16:42.811920+03:00	2
7	Turkey	very beaut...	2021-10-31 23:24:41.961766+02:00	5

Choose option:

PRESS: 1 to add... 2 to update... 3 to delete... 4 to specific\_select... 5 to show\_table...

2

Choose table:

press 1 - users... press 2 - posts... press 3 - likes... press 4 - comments

2

How much rows do you wanna update?

1

Enter values following this sequence: title, body, author\_id

title: Edited post

body: fsdfsdfsd

author\_id: 5

post\_id :1

Successfully updated

Do you want to continue? Press Y if yes, N if no: |

post_id	title	body	date	author_id
1	Edited post	fsdfsdfsd	2021-09-11 14:16:42.811920+03:00	5
2	New PC	So last week i bought a new PC....	2021-09-11 14:17:45.551787+03:00	2
4	My last job	fsdfsdfsdksdfdsfksdfsd	2021-09-11 14:18:28.289647+03:00	3
5	Msdksdafsd	sfsadfsd	2021-10-31 16:16:09.707603+02:00	1
6	fsdf	fsdffd	2021-10-31 21:49:22.683802+02:00	3
7	Turkey	very beaut...	2021-10-31 23:24:41.961766+02:00	5
8	f499491fa6658cfd0b239dbb39b16f00	699ae7e2693a253ee271ac1f8de00f5e	2021-10-31 23:24:41.961766+02:00	3
9	1a044eda4a44bc7accaedecd4a9990f1	53a41b941e30ef2cae9b8457c1ca9371	2021-10-31 23:24:41.961766+02:00	3
1164019	Good memories	sdfksdafksdsdafsd	2021-12-23 13:50:41.004283+02:00	1

Enter values following this sequence: title, body, author\_id

title: fsdffd

body: fsdfsdf

author\_id: 100

post\_id :1

Failed updating record of the table {} (psycopg2.errors.ForeignKeyViolation) insert or update on table "posts" violates foreign key constraint: Key (author\_id)=(100) is not present in table "users".

[SQL: UPDATE posts SET author\_id=%(author\_id)s, title=%(title)s, body=%(body)s WHERE posts.post\_id = %(post\_id\_1)s]

[parameters: {'author\_id': '100', 'title': 'fsdffd', 'body': 'fsdfsdf', 'post\_id\_1': '1'}]

(Background on this error at: <https://sqlalche.me/e/14/gkpi>)

Do you want to exit? Press M to go to the main menu, or E to exit: |

## Table Likes:

like_id	user_id	post_id
101020	3	1
101025	5	4

Enter values following this sequence: user\_id, post\_id

user\_id: 1

post\_id: 1

like\_id :101020

Successfully updated

Do you want to continue? Press Y if yes, N if no: |

like_id	user_id	post_id
101020	1	1
101025	5	4

## Error handling:

Enter values following this sequence: user\_id, post\_id

user\_id: 3

post\_id: 3

like\_id :101020

Failed updating record of the table {} (psycopg2.errors.ForeignKeyViolation) insert or update on table "likes" violates foreign key constraint  
DETAIL: Key (post\_id)=(3) is not present in table "posts".

[SQL: UPDATE likes SET user\_id=%(user\_id)s, post\_id=%(post\_id)s WHERE likes.like\_id = %(like\_id\_1)s]

[parameters: {'user\_id': '2', 'post\_id': '3', 'like\_id\_1': '101020'}]

(Background on this error at: <https://sqlalche.me/e/14/gkpf>)

Do you want to exit? Press M to go to the main menu, or E to exit: |

## Table: comments

comment_id	body	date	author_id	post_id
100020	Nice info	2021-12-23 14:16:42.763118+02:00	1	4

Enter values following this sequence: body, author\_id, post\_id comment\_id

body: *Good ph*

user\_id: *2*

post\_id: *4*

comment\_id :*100020*

Successfully updated

Do you want to continue? Press Y if yes, N if no:

comment_id	body	date	author_id	post_id
100020	Good ph	2021-12-23 14:16:42.763118+02:00	2	4

## Error handling

```
Enter values following this sequence: body, author_id, post_id comment_id
body: dsfds
user_id: 1
post_id: 103
comment_id :100020
Failed updating record of the table {} (psycopg2.errors.ForeignKeyViolation) insert or update on table "comments" violates foreign key constraint
DETAIL:  Key (post_id)=(103) is not present in table "posts".

[SQL: UPDATE comments SET author_id=%(author_id)s, post_id=%(post_id)s, body=%(body)s WHERE comments.comment_id = %(comment_id_1)s]
[parameters: {'author_id': '1', 'post_id': '103', 'body': 'dsfds', 'comment_id_1': '100020'}]
(Background on this error at: https://sqlalche.me/e/14/gkpi)
Do you want to exit? Press M to go to the main menu, or E to exit:
```

## Deleting

### Table posts:

post_id	title	body	date	author_id
1	Edited post	fsdfsd fsdfs	2021-09-11 14:16:42.811920+03:00	5
2	New PC	So last week i bought a new PC...	2021-09-11 14:17:45.551787+03:00	2
4	My last job	fsdfsd fsdkfsdkfsdlfksd fsdfs	2021-09-11 14:18:28.289647+03:00	3
5	Msdkfsdafsd	sfdsad fsdfsdf	2021-10-31 16:16:09.707603+02:00	1
6	fsdf	fsdffsd	2021-10-31 21:49:22.683802+02:00	3
7	Turkey	very beaut...	2021-10-31 23:24:41.961766+02:00	5
8	f499491fa6658cfd0b239dbb39b16f00	699ae7e2693a253ee271ac1f8de00f5e	2021-10-31 23:24:41.961766+02:00	3
9	1a044eda4a44bc7accaedecd4a9990f1	53a41b941e30ef2cae9b8457c1ca9371	2021-10-31 23:24:41.961766+02:00	3
1164019	Good memories	sfd fksdafksdsdafsd fsd	2021-12-23 13:50:41.004283+02:00	1



```

PRESS: 1 to add... 2 to update... 3 to delete... 4 to specific_select... 5 to show_table...
3
Choose table:
press 1 - users... press 2 - posts... press 3 - likes... press 4 - comments
2
How much rows do you wanna delete?
1
Enter value that marks post_id: 8
Successfully deleted

```

post_id	title	body	date	author_id
1	Edited post	fsdfsdfsdfs	2021-09-11 14:16:42.811920+03:00	5
2	New PC	So last week i bought a new PC...	2021-09-11 14:17:45.551787+03:00	2
4	My last job	fsdfsdfsdkfsdkfslfkfsdfsdfs	2021-09-11 14:18:28.289647+03:00	3
5	Msdksdafsd	sfsadfsdfsdf	2021-10-31 16:16:09.707603+02:00	1
6	fsdf	fsdffsd	2021-10-31 21:49:22.683802+02:00	3
7	Turkey	very beaut...	2021-10-31 23:24:41.961766+02:00	5
9	1a044eda4a44bc7accaedecd4a9990f1	53a41b941e30ef2cae9b8457c1ca9371	2021-10-31 23:24:41.961766+02:00	3
1164019	Good memories	sfdkksdafksdsdafsd	2021-12-23 13:50:41.004283+02:00	1

## Error handling:

```

How much rows do you wanna delete?
1
Enter value that marks post_id: 1
Failed deleting record into table (psycopg2.errors.ForeignKeyViolation) update or delete on table "posts" violates foreign key const
DETAIL: Key (post_id)=(1) is still referenced from table "likes".

[SQL: DELETE FROM posts WHERE posts.post_id = %(post_id)s]
[parameters: {'post_id': 1}]
(Background on this error at: https://sqlalche.me/e/14/gkqpj)
Do you want to continue? Press Y if yes, N if no:

```

## Table users:

user_id	f_name	l_name	email	password
1	1	fsdfsdf	fsdfsdf	sdfsdfsdf
2	2	Danes	johndanes@gmail.com	qwerty
3	3	Banes	darek@gmail.com	zxcvbn
5	5	аыбацук	zenya@gmail.com2	аыбаыб
6	6	fsdfsdf	123214sa	fsdffd
8	8	Vasiluk	igor@gmail.com	flsdfkwe
9	9	SHKCWL	YKAJL	HNPSKL
452848	452848	fsdafsdafsd	safdsfas	sdfasdfsdf
452850	452850	Darrinson	dskfsdfs@gmail.com	dsfkfsdfsdf
452851	452851	ddfsd	kfkssdmf@gmail.com	1

```
Enter value that marks user_id:452851
Successfully deleted
```

user_id	f_name	l_name	email	password
1	1	fsdfsdf	fsdfsdf	sdfsdfsdf
2	2	Danes	johndanes@gmail.com	qwerty
3	3	Banes	darek@gmail.com	zxcvbn
5	5	аыВацук	zenya@gmail.com2	аыыаыы
6	6	fsdfsdf	123214sa	fsddfsd
8	8	Vasiluk	igor@gmail.coom	flsdfkwe
9	9	SHKCWL	YKAJL	HNPSKL
452848	452848	fsdafsdafsd	safdsfas	sdfasdfsdf
452850	452850	Darrinson	dskfsdfs@gmail.com	dsfkfsdfsdf

## Error handling:

```
Choose option:
PRESS: 1 to add... 2 to update... 3 to delete... 4 to specific_select... 5 to show_table...
└─┘
Choose table:
press 1 - users... press 2 - posts... press 3 - likes... press 4 - comments
└─┘
How much rows do you wanna delete?
└─┘
Enter value that marks user_id:
Failed deleting record into table (psycopg2.errors.ForeignKeyViolation) update or delete on table "users" violates foreign key constraint
DETAIL: Key (user_id)=(1) is still referenced from table "posts".

[SQL: DELETE FROM users WHERE users.user_id = %(user_id)s]
[parameters: {'user_id': 1}]
(Background on this error at: https://sqlalche.me/e/14/gkpi)
Do you want to continue? Press Y if yes, N if no: |
```

## Базові класи таблиць в SQLAlchemy ORM

```
class Comment(Model.Base):
    __tablename__ = 'comments'
    comment_id = Column(Integer, primary_key=True)
    author_id = Column(Integer, ForeignKey('users.user_id'))
    post_id = Column(Integer, ForeignKey('posts.post_id'))
    body = Column(String)
    date = Column(DateTime, default=datetime.datetime.utcnow)

    user = relationship("User")
    post = relationship("Post")

    def __init__(self, body, author_id, post_id):
        self.body = body
        self.author_id = author_id
        self.post_id = post_id
```

```
class Like(Model.Base):
    __tablename__ = 'likes'
    like_id = Column(Integer, primary_key=True)
    user_id = Column(Integer, ForeignKey('users.user_id'))
    post_id = Column(Integer, ForeignKey('posts.post_id'))

    user = relationship("User")
    post = relationship("Post")

    def __init__(self, user_id, post_id):
        self.user_id = user_id
        self.post_id = post_id
```

```
class Post(Model.Base):
    __tablename__ = 'posts'
    post_id = Column(Integer, primary_key=True)
    author_id = Column(Integer, ForeignKey('users.user_id'))
    title = Column(String)
    body = Column(String)
    date = Column(DateTime, default=datetime.datetime.utcnow)

    user = relationship("User")

    def __init__(self, author_id, title, body):
        self.author_id = author_id
        self.title = title
        self.body = body
```

```
class User(Model.Base):
    __tablename__ = 'users'
    user_id = Column(Integer, primary_key=True)
    f_name = Column(String)
    l_name = Column(String)
    email = Column(String)
    password = Column(String)

    def __init__(self, f_name, l_name, email, password):
        self.f_name = f_name
        self.l_name = l_name
        self.email = email
        self.password = password
```

## Завдання 2

### BTree

Створимо таблицю для тестування:

```
drop table if exists btree_test;  
create table btree_test(id bigserial Primary key, count integer);  
insert into btree_test(count) select random()*999999 from generate_series(1, 1000000) as q
```

### Без індексування:

```
explain analyze  
select count(*) from btree_test where id % 3 =0
```

Output Explain Messages Notifications

QUERY PLAN	
text	
Finalize Aggregate (cost=12661.42..12661.43 rows=1 width=8) (actual time=82.458..85.541 rows=1 loops=1)	
[...] -> Gather (cost=12661.21..12661.42 rows=2 width=8) (actual time=82.444..85.531 rows=3 loops=1)	
[...] Workers Planned: 2	
[...] Workers Launched: 2	
[...] -> Partial Aggregate (cost=11661.21..11661.22 rows=1 width=8) (actual time=77.592..77.594 rows=1 loops=3)	
[...] -> Parallel Seq Scan on btree_test (cost=0.00..11656.00 rows=2083 width=0) (actual time=0.011..69.516 rows=111111 loops=3)	
[...] Filter: ((id % '3'::bigint) = 0)	
[...] Rows Removed by Filter: 222222	
Planning Time: 0.079 ms	
Execution Time: 85.593 ms	

```
-- drop index if exists btree_idx;  
--create index btree_idx on btree_test using btree(count)  
explain analyze  
select count(count) from btree_test where count=500000;
```


Output Explain Messages Notifications

QUERY PLAN	
text	
Aggregate (cost=11614.54..11614.55 rows=1 width=8) (actual time=66.589..71.941 rows=1 loops=1)	
[...] -> Gather (cost=1000.00..11614.53 rows=2 width=4) (actual time=66.582..71.933 rows=0 loops=1)	
[...] Workers Planned: 2	
[...] Workers Launched: 2	
[...] -> Parallel Seq Scan on btree_test (cost=0.00..10614.33 rows=1 width=4) (actual time=59.940..59.941 rows=0 loops=3)	
[...] Filter: (count = 500000)	
[...] Rows Removed by Filter: 333333	
Planning Time: 0.213 ms	
Execution Time: 71.991 ms	

explain analyze

select avg(count) from btree\_test where count=4989 and count = 187001

Output Explain Messages Notifications

QUERY PLAN	
text	
Aggregate (cost=11614.44..11614.45 rows=1 width=32) (actual time=4.972..8.071 rows=1 loops=1)	
[...] -> Gather (cost=1000.00..11614.43 rows=1 width=4) (actual time=4.966..8.062 rows=0 loops=1)	
[...] Workers Planned: 2	
[...] Workers Launched: 2	
[...] -> Result (cost=0.00..10614.33 rows=1 width=4) (actual time=0.002..0.002 rows=0 loops=3)	
[...] One-Time Filter: false	
[...] -> Parallel Seq Scan on btree_test (cost=0.00..10614.33 rows=1 width=4) (never executed)	
[...] Filter: (count = 4989)	
Planning Time: 0.070 ms	
Execution Time: 8.112 ms	

explain analyze

select max(count) from btree\_test where count between 40000 and 60000 group by 1d % 3 = 0

Output Explain Messages Notifications

QUERY PLAN	
text	
Finalize GroupAggregate (cost=12741.01..12741.53 rows=2 width=5) (actual time=71.139..77.299 rows=2 loops=1)	
[...] Group Key: (((id % '3'::bigint) = 0))	
[...] -> Gather Merge (cost=12741.01..12741.48 rows=4 width=5) (actual time=71.130..77.289 rows=6 loops=1)	
[...] Workers Planned: 2	
[...] Workers Launched: 2	
[...] -> Sort (cost=11740.99..11740.99 rows=2 width=5) (actual time=64.335..64.338 rows=2 loops=3)	
[...] Sort Key: (((id % '3'::bigint) = 0))	
[...] Sort Method: quicksort Memory: 25kB	
[...] Worker 0: Sort Method: quicksort Memory: 25kB	
[...] Worker 1: Sort Method: quicksort Memory: 25kB	
[...] -> Partial HashAggregate (cost=11740.95..11740.98 rows=2 width=5) (actual time=64.282..64.285 rows=2 loops=3)	
[...] Group Key: ((id % '3'::bigint) = 0)	
[...] Batches: 1 Memory Usage: 24kB	
[...] Worker 0: Batches: 1 Memory Usage: 24kB	
[...] Worker 1: Batches: 1 Memory Usage: 24kB	
[...] -> Parallel Seq Scan on btree_test (cost=0.00..11698.48 rows=8495 width=5) (actual time=0.016..62.079 rows=6680 loops=3)	
[...] Filter: ((count >= 40000) AND (count <= 60000))	
[...] Rows Removed by Filter: 326654	
Planning Time: 0.137 ms	
Execution Time: 77.340 ms	

### З індексуванням

**explain analyze**

**select count(\*) from btree\_test where id % 3 =0;**

[Output](#) [Explain](#) [Messages](#) [Notifications](#)

**QUERY PLAN**

text



Finalize Aggregate (cost=12661.42..12661.43 rows=1 width=8) (actual time=71.938..77.618 rows=1 loops=1)

[...] -> Gather (cost=12661.21..12661.42 rows=2 width=8) (actual time=71.830..77.606 rows=3 loops=1)

[...] Workers Planned: 2

[...] Workers Launched: 2

[...] -> Partial Aggregate (cost=11661.21..11661.22 rows=1 width=8) (actual time=67.994..67.996 rows=1 loops=3)

[...] -> Parallel Seq Scan on btree\_test (cost=0.00..11656.00 rows=2083 width=0) (actual time=0.018..57.928 rows=111111 loops=3)

[...] Filter: ((id % '3'::bigint) = 0)

[...] Rows Removed by Filter: 222222

Planning Time: 0.107 ms

Execution Time: 77.657 ms

```
-- drop index if exists btree_idx;  
-- create index btree_idx on btree_test using btree(count)  
explain analyze  
select count(count) from btree_test where count=500000;
```

[a Output](#) [Explain](#) [Messages](#) [Notifications](#)

**QUERY PLAN**

text



Aggregate (cost=4.46..4.47 rows=1 width=8) (actual time=0.026..0.027 rows=1 loops=1)

[...] -> Index Only Scan using btree\_idx on btree\_test (cost=0.42..4.46 rows=2 width=4) (actual time=0.021..0.021 rows=0 loops=1)

[...] Index Cond: (count = 500000)

[...] Heap Fetches: 0

Planning Time: 0.313 ms

Execution Time: 0.056 ms

```
-- drop index if exists btree_idx;  
-- create index btree_idx on btree_test using btree(count)  
explain analyze  
select avg(count) from btree_test where count=4989 and count = 187001
```

[a Output](#) [Explain](#) [Messages](#) [Notifications](#)

**QUERY PLAN**

text



Aggregate (cost=4.46..4.47 rows=1 width=32) (actual time=0.005..0.006 rows=1 loops=1)

[...] -> Result (cost=0.42..4.46 rows=1 width=4) (actual time=0.001..0.001 rows=0 loops=1)

[...] One-Time Filter: false

[...] -> Index Only Scan using btree\_idx on btree\_test (cost=0.42..4.46 rows=1 width=4) (never executed)

[...] Index Cond: (count = 4989)

[...] Heap Fetches: 0

Planning Time: 0.336 ms

Execution Time: 0.043 ms

```
-- drop index if exists btree_idx;
-- create index btree_idx on btree_test using btree(count)
explain analyze
select max(count) from btree_test where count between 40000 and 60000 group by id % 3 = 0
```

Output Explain Messages Notifications

QUERY PLAN	
text	
HashAggregate (cost=6325.10..6325.13 rows=2 width=5) (actual time=19.708..19.710 rows=2 loops=1)	
[...] Group Key: ((id % '3'::bigint) = 0)	
[...] Batches: 1 Memory Usage: 24kB	
[...] -> Bitmap Heap Scan on btree_test (cost=409.40..6223.16 rows=20388 width=5) (actual time=3.914..16.552 rows=20039 loops=1)	
[...] Recheck Cond: ((count >= 40000) AND (count <= 60000))	
[...] Heap Blocks: exact=5282	
[...] -> Bitmap Index Scan on btree_idx (cost=0.00..404.31 rows=20388 width=0) (actual time=3.104..3.105 rows=20039 loops=1)	
[...] Index Cond: ((count >= 40000) AND (count <= 60000))	
Planning Time: 0.184 ms	
Execution Time: 19.745 ms	

З результатів виконання даних запитів, можемо побачити, що Btree є дуже ефективним для даних які можливо відсортувати, тобто для тих, до яких можна застосувати оператори порівняння.

## GIN

GIN розшифровується як узагальнений інверсований індекс. В нього індексуються не самі значення, а окремі елементи; кожен елемент посилається на те значення, у яких він зустрічається. До кожного елемента прив'язується упорядкований набір посилань на рядки таблиць, що містять значення з цим елементом. Елементи ніколи не видаляються з GIN-індексу. Таке рішення істотно спрощує алгоритми, забезпечуючи паралельну роботу з індексом кількох процесів.

Якщо список TID(tuple identifier) невеликий, то він розміщується на тій же сторінці, що і елемент (і називається список розміщення). В іншому випадку для ефективної структури даних використовується В-дерево.

Недоліком даного методу є те, що вставка або ж оновлення даних виконується повільно у зв'язку з великою кількістю лексем, які необхідно проіндексувати. Перевагою є гарна компактність. Одна й та ж лексема зберігається завжди один раз. Також TID зберігається в індексі впорядковано, а це дає можливість використовувати стискання: кожен



наступний у списку TID зберігається як різниця з попереднім - зазвичай це невелике число, на яке потрібно набагато менше бітів, ніж на повний TID

```
INSERT INTO posts (title, body, author_id)
  SELECT ttl, bdy, author_id FROM
  (SELECT md5((random()*1)::text) as ttl,
        md5((random()*2)::text) as bdy,
        author_id
  FROM posts tablesample BERNOULLI(100)
  ORDER BY random()) k, generate_series(1, 100000) LIMIT 900000
```

INSERT 0 900000

Query returned successfully in 14 secs 468 msec.

```
1 select count(*) from posts
```

Data Output Explain Messages Notifications

	count bigint	
1	900010	

```
1 explain select * from posts
```

Data Output Explain Messages Notifications

	QUERY PLAN	
	text	
1	Seq Scan on posts (cost=0.00..21858.10 rows=900010 width=82)	

```
1 CREATE EXTENSION IF NOT EXISTS pg_trgm;
2 CREATE INDEX gin_idx ON posts USING gin(title gin_trgm_ops);
```

Data Output Explain Messages Notifications

CREATE INDEX

Query returned successfully in 8 secs 738 msec.

Як можна побачити редактор обирає найбільш оптимальний варіант для

кожного випадку. За умов показаних нижче використовується послідовний пошук, оскільки більша частина значень підпадає під цей пошук.

```
1 explain analyze SELECT * FROM posts where title ilike '%b%';
```

Data Output Explain Messages Notifications

	QUERY PLAN	
	text	
1	Seq Scan on posts (cost=0.00..24108.12 rows=722558 width=82) (actual time...	
2	[...] Filter: ((title)::text ~* '%b% '::text)	
3	[...] Rows Removed by Filter: 180007	
4	Planning Time: 0.983 ms	
5	Execution Time: 802.866 ms	

```
1 SELECT count(*) FROM posts where title ilike '%bf%'
```

Data Output Explain Messages Notifications

	count	
	bigint	
1	90000	

В даному випадку виконується паралельний послідовний пошук, оскільки GIN індекс не завжди може бути сумісним із таким видом пошуку. Так як індексація відбувається не по значенню, а по окремим частинам цього елемента.

```
1 explain analyze
2 SELECT count(*) FROM posts where title ilike '%bf%'
```

Data Output Explain Messages Notifications

	QUERY PLAN	
	text	
1	Finalize Aggregate (cost=18642.67..18642.68 rows=1 width=8) (actual time=5...	
2	[...] -> Gather (cost=18642.46..18642.67 rows=2 width=8) (actual time=563.30...	
3	[...] Workers Planned: 2	
4	[...] Workers Launched: 2	
5	[...] -> Partial Aggregate (cost=17642.46..17642.47 rows=1 width=8) (actual ti...	
6	[...] -> Parallel Seq Scan on posts (cost=0.00..17545.55 rows=38763 width=0) ...	
7	[...] Filter: ((title)::text ~* '%bf% '::text)	
8	[...] Rows Removed by Filter: 270003	
9	Planning Time: 0.157 ms	
10	Execution Time: 573.884 ms	

Далі використовується індексний gin-пошук. Він працює наступним чином. Спочатку пошуковий запит виділяє лексеми(ключі пошуку): 'f1%'('a%'). Далі з допомогою B-дерева або сторінки за ключем відбувається перебір готових TID списків. Після чого для кожного знайденого TID списку виконується спеціальна функція, яка визначає значення, які підпадають під пошуковий запит. А bitmap scan використовується через те, що результат повертається у вигляді бітової карти. В результаті пошук по індексу показує кращу швидкодію.

```
1 explain analyze
2 SELECT * FROM posts where title ilike 'f1%' or title ilike 'a%'
```

Data Output Explain Messages Notifications

	QUERY PLAN	
	text	
1	Bitmap Heap Scan on posts (cost=975.61..15212.87 rows=91951 width=82) (a...	
2	[...] Recheck Cond: (((title)::text ~~* 'f1% '::text) OR ((title)::text ~~* 'a% '::text))	
3	[...] Heap Blocks: exact=12858	
4	[...] -> BitmapOr (cost=975.61..975.61 rows=91951 width=0) (actual time=16.0...	
5	[...] -> Bitmap Index Scan on gin_idx (cost=0.00..100.00 rows=1 width=0) (actu...	
6	[...] Index Cond: ((title)::text ~~* 'f1% '::text)	
7	[...] -> Bitmap Index Scan on gin_idx (cost=0.00..829.63 rows=91951 width=0) ...	
8	[...] Index Cond: ((title)::text ~~* 'a% '::text)	
9	Planning Time: 0.119 ms	
10	Execution Time: 272.505 ms	

```
1 explain analyze
2 SELECT * FROM posts where title ilike 'f1%' -- or title ilike 'a3%'
```

Data Output Explain Messages Notifications

	QUERY PLAN	
	text	
1	Bitmap Heap Scan on posts (cost=100.00..104.01 rows=1 width=82) (actual ti...	
2	[...] Recheck Cond: ((title)::text ~~* 'f1% '::text)	
3	[...] -> Bitmap Index Scan on gin_idx (cost=0.00..100.00 rows=1 width=0) (actu...	
4	[...] Index Cond: ((title)::text ~~* 'f1% '::text)	
5	Planning Time: 0.115 ms	
6	Execution Time: 0.043 ms	

```

1 explain analyze
2 SELECT count(*) FROM posts where title ilike 'f1%' -- or title ilike 'a3%'

```

Data Output Explain Messages Notifications

	QUERY PLAN	
	text	
1	Aggregate (cost=104.02..104.03 rows=1 width=8) (actual time=0.030..0.031 ro...	
2	[...] -> Bitmap Heap Scan on posts (cost=100.00..104.01 rows=1 width=0) (actu...	
3	[...] Recheck Cond: ((title)::text ~~~* 'f1%':text)	
4	[...] -> Bitmap Index Scan on gin_idx (cost=0.00..100.00 rows=1 width=0) (actu...	
5	[...] Index Cond: ((title)::text ~~~* 'f1%':text)	
6	Planning Time: 0.173 ms	
7	Execution Time: 0.080 ms	

```

1 explain analyze
2 SELECT count(*) FROM posts where title ilike 'f1%' group by author_id -- 'f1%' -- or title ilike 'a3%'

```

Data Output Explain Messages Notifications

	QUERY PLAN	
	text	
1	GroupAggregate (cost=108.02..108.04 rows=1 width=12) (actual time=0.041.....	
2	[...] Group Key: author_id	
3	[...] -> Sort (cost=108.02..108.03 rows=1 width=4) (actual time=0.039..0.041 r...	
4	[...] Sort Key: author_id	
5	[...] Sort Method: quicksort Memory: 25kB	
6	[...] -> Bitmap Heap Scan on posts (cost=104.00..108.01 rows=1 width=4) (actu...	
7	[...] Recheck Cond: ((title)::text ~~~* 'f1%':text)	
8	[...] -> Bitmap Index Scan on gin_idx (cost=0.00..104.00 rows=1 width=0) (actu...	
9	[...] Index Cond: ((title)::text ~~~* 'f1%':text)	
10	Planning Time: 0.191 ms	
11	Execution Time: 0.104 ms	

```

1 explain analyze
2 SELECT * FROM posts where title ilike 'f1%' or title ilike 'a%' order by author_id desc

```

Data Output Explain Messages Notifications

	QUERY PLAN	
	text	
1	Sort (cost=27204.58..27434.46 rows=91951 width=82) (actual time=309.075....	
2	[...] Sort Key: author_id DESC	
3	[...] Sort Method: external merge Disk: 8296kB	
4	[...] -> Bitmap Heap Scan on posts (cost=983.61..15220.87 rows=91951 width...	
5	[...] Recheck Cond: (((title)::text ~~~* 'f1%':text) OR ((title)::text ~~~* 'a%':text))	
6	[...] Heap Blocks: exact=12858	
7	[...] -> BitmapOr (cost=983.61..983.61 rows=91951 width=0) (actual time=18.1...	
8	[...] -> Bitmap Index Scan on gin_idx (cost=0.00..104.00 rows=1 width=0) (actu...	
9	[...] Index Cond: ((title)::text ~~~* 'f1%':text)	
10	[...] -> Bitmap Index Scan on gin_idx (cost=0.00..833.63 rows=91951 width=0) ...	
11	[...] Index Cond: ((title)::text ~~~* 'a%':text)	
12	Planning Time: 0.228 ms	
13	Execution Time: 331.017 ms	

## Завдання 3

*after insert,  
update*

Створюємо таблицю для тестування тригера

```
DROP TABLE IF EXISTS "test_table_1";
CREATE TABLE "test_table_1"("id" bigserial PRIMARY KEY, "Text" text);

DROP TABLE IF EXISTS "test_table_2";
CREATE TABLE "test_table_2"("id" bigserial PRIMARY KEY, "ID" bigint, "Text" text);

INSERT INTO "test_table_1"("Text")
VALUES ('Text1'), ('Text2'), ('Text3'), ('Text4'), ('Text5'), ('Text6'), ('Text7'), ('Text8'), ('Text9'), ('Text10');
```

## Створення тригера

```
1 CREATE OR REPLACE FUNCTION after_insert_update_func()
2 RETURNS TRIGGER AS $$
3 DECLARE
4     CURSOR_LOG CURSOR FOR SELECT * FROM "test_table_2";
5     row_ "test_table_2"%ROWTYPE;
6
7 BEGIN
8     IF new."id" % 2 = 0 THEN
9         RAISE NOTICE 'id is multiple on 2'; FOR row_ IN CURSOR_LOG LOOP
10             UPDATE "test_table_2" SET "ID" = row_."id" WHERE "id" = row_."id";
11             END LOOP; RETURN OLD;
12     ELSE
13         RAISE NOTICE 'id is not multiple on 2';
14         Insert into "test_table_2" ("ID","Text") values (new."id", 'new_Text');
15         RETURN NEW;
16     END IF;
17 END;
18 $$
19 LANGUAGE PLPGSQL
```

```
CREATE TRIGGER "after_insert_update_trigger"
AFTER INSERT OR UPDATE ON "test_table_1"
FOR EACH ROW
EXECUTE procedure after_insert_update_func();
```

Початковий стан

	Id [PK] bigint	Text text
1	1	Text1
2	2	Text2
3	3	Text3
4	4	Text4
5	5	Text5
6	6	Text6
7	7	Text7
8	8	Text8
9	9	Text9
10	10	Text10

	Id [PK] bigint	ID bigint	Text text

Після вставки

1

INSERT INTO "test\_table\_1"("Text") VALUES ('New text1');

Data Output

Explain

Messages

Notifications

NOTICE: id is not multiple on 2

INSERT 0 1

Query returned successfully in 309 msec.

1

select \* from test\_table\_1;

2

-- select \* from test\_table\_2;

Data Output

Explain

Messages

Notifications

	Id [PK] bigint	Text text
1	1	Text1
2	2	Text2
3	3	Text3
4	4	Text4
5	5	Text5
6	6	Text6
7	7	Text7
8	8	Text8
9	9	Text9
10	10	Text10
11	11	New text1

2

select \* from test\_table\_2;

Data Output

Explain

Messages

Notifications

	Id [PK] bigint	ID bigint	Text text
1	1	11	new_Text

## Повторна вставка, вже парне id

```
1 insert into "test_table_1"("Text") values('New text 2')
```

Data Output Explain Messages Notifications

NOTICE: id is multiple on 2  
INSERT 0 1

```
1 select * from test_table_1
```

Data Output Explain Messages Notif

	Id [PK] bigint	Text text
1	1	Text1
2	2	Text2
3	3	Text3
4	4	Text4
5	5	Text5
6	6	Text6
7	7	Text7
8	8	Text8
9	9	Text9
10	10	Text10
11	11	New text1
12	12	New text 2

```
1 select * from test_table_2
```

Data Output Explain Messages N

	Id [PK] bigint	ID bigint	Text text
1	1	1	new_Text

Можемо побачити, що виконалась інша гілка коду, у випадку коли id ділиться націло на 2

Після update:

```
1 update "test_table_1" set "Text" = 'NEW_TEXT' where "id"= 11;
```

Data Output Explain Messages Notifications

NOTICE: id is not multiple on 2  
UPDATE 1

Query returned successfully in 141 msec.

Data Output Explain Messages

	Id [PK] bigint	Text text
1	1	Text1
2	2	Text2
3	3	Text3
4	4	Text4
5	5	Text5
6	6	Text6
7	7	Text7
8	8	Text8
9	9	Text9
10	10	Text10
11	11	NEW_TEXT
12	12	New text 2

	Id [PK] bigint	ID bigint	Text text
1	1	1	new_Text
2	2	11	new_Text



## Завдання 4

```
DROP TABLE IF EXISTS "table_transactions";
CREATE TABLE table_transactions(
    id bigserial PRIMARY KEY,
    text text,
    num bigint
);
```

```
INSERT INTO "table_transactions"(text, num) VALUES ('Text1', 1), ('Text2', 2), ('Text4', 4);
```

### READ COMMITTED

На цьому рівні ізоляції одна транзакція не бачить змін у базі даних, викликаних іншою доки та не завершить своє виконання (командою COMMIT або ROLLBACK).

```
blog=# start transaction;
START TRANSACTION
blog=# set transaction isolation level read committed read write;
SET
blog=# update table_transactions set num=3;
UPDATE 3
blog=# select * from table_transactions;
 id | text  | num
-----+-----+-----
  7 | Text1 |   3
  8 | Text2 |   3
  9 | Text4 |   3
(3 рядки)
```

```
psql (14.1 (Ubuntu 14.1-2.pgdg20.04+1), сервер 13.5 (Ubuntu 13.5-2.pgdg20.04+1))
Ви тепер під'єднані до бази даних "blog" як користувач "zenya".
blog=# select * from table_transactions;
 id | text  | num
-----+-----+-----
  7 | Text1 |   1
  8 | Text2 |   2
  9 | Text4 |   4
(3 рядки)
blog=#
```

```
blog=# start transaction;
START TRANSACTION
blog=# set transaction isolation level read committed read write;
SET
blog=# update table_transactions set num=3;
UPDATE 3
blog=# select * from table_transactions;
 id | text  | num
-----+-----+-----
  7 | Text1 |   3
  8 | Text2 |   3
  9 | Text4 |   3
(3 рядки)
blog=# commit;
COMMIT
```

```
Ви тепер під'єднані до бази даних "blog" як користувач "zenya".
blog=# select * from table_transactions;
 id | text  | num
-----+-----+-----
  7 | Text1 |   1
  8 | Text2 |   2
  9 | Text4 |   4
(3 рядки)
blog=# select * from table_transactions;
 id | text  | num
-----+-----+-----
  7 | Text1 |   3
  8 | Text2 |   3
  9 | Text4 |   3
(3 рядки)
```

Бачимо, що до моменту, коли ми не завершили виконання транзакції, командою commit, зміни не відображаються у базі даних.

Аналогічна ситуація для вставки та видалення даних

```

blog=# start transaction;
START TRANSACTION
blog=# set transaction isolation level read committed read write;
SET
blog=# delete from table_transactions where id = 7;
DELETE 1
blog=# insert into table_transactions("text", num) values ('some tex', 5);
INSERT 0 1
blog=# commit
blog-## ;
COMMIT
blog=# 

```

```

blog=# select * from table_transactions;
 id | text | num
-----+-----+-----
  7 | Text1 |   3
  8 | Text2 |   3
  9 | Text4 |   3
(3 рядки)

```

```

blog=# select * from table_transactions;
 id | text | num
-----+-----+-----
  8 | Text2 |   3
  9 | Text4 |   3
 10 | some tex |   5
(3 рядки)

```

Також можемо побачити, що допоки не завершилась одна транзакція, інша не може бути виконана.

```

blog=# start transaction;
START TRANSACTION
blog=# update table_transactions set num=3;
UPDATE 3
blog=# 

```

```

blog=# update table_transactions set num=3;

```

Лише після закінчення лівої транзакції, запит виконала друга, змінивши дані, знову. При чому дані в першій транзакції після коміта зміняться перший раз і потім ще раз після коміта другої транзакції, що і зображено на скріншоті.

```

blog=# update table_transactions set num=1;
UPDATE 3
blog=# commit
blog-## ;
COMMIT
blog=# select * from table_transactions;
 id | text | num
-----+-----+-----
  8 | Text2 |   1
  9 | Text4 |   1
 10 | some tex |   1
(3 рядки)

```

```

blog=# select * from table_transactions;
 id | text | num
-----+-----+-----
  8 | Text2 |   2
  9 | Text4 |   2
 10 | some tex |   2
(3 рядки)

```

```

blog=# update table_transactions set num=2;
UPDATE 3
blog=# select * from table_transactions;
 id | text | num
-----+-----+-----
  8 | Text2 |   2
  9 | Text4 |   2
 10 | some tex |   2
(3 рядки)

```

```

blog=# commit;
COMMIT
blog=# select * from table_transactions;
 id | text | num
-----+-----+-----
  8 | Text2 |   2
  9 | Text4 |   2
 10 | some tex |   2
(3 рядки)

```

Отже, коли транзакція 2 бачить дані транзакції 1, запитів UPDATE, DELETE виникає феномен повторного читання, а коли бачить дані запиту INSERT – читання фантомів. Цей рівень ізоляції забезпечує захист від явища брудного читання.

## REPEATABLE READ

На цьому рівні ізоляції транзакція 2 не бачитиме змінені дані транзакцією 1, але також не може отримати доступ до тих самих даних.

На даному скріншоті можна переконатися в тому, що друга транзакція не бачить змін в першій транзакції.

```
blog=# start transaction;
START TRANSACTION
blog=# set transaction isolation level repeatable read read write;
SET
blog=# delete from table_transactions where id = 9
blog-## ;
DELETE 1
blog=# commit;
COMMIT
blog=#

blog=# select * from table_transactions;
 id |  text  | num
-----+-----
  8 | Text2  |  2
  9 | Text4  |  2
 10 | some tex |  2
(3 рядки)

blog=# delete from table_transactions where id = 9;
ERROR:  could not serialize access due to concurrent delete
blog-!#
```

При спробі доступу будемо отримувати помилку, що також зображено на скріншоті.

При такій ізоляції, не виникає читання фантомів та повторного читання, а також заборонено одночасний доступ до не збережених даних. Хоча класично цей рівень ізоляції призначений для попередження повторного читання.

## SERIALIZABLE

На цьому рівні транзакції поведуть себе так, ніби вони не знають одна про одну. Вони не можуть вплинути одна на одну і одночасний доступ строго заборонений.

```
blog=# start transaction;;
START TRANSACTION
blog=# set transaction isolation level serializable read write;
SET
blog=# delete from table_transactions where id = 8;
DELETE 1
blog=# select * from table_transactions;
 id |  text  | num
-----+-----
 10 | some tex |  2
(1 рядок)

blog=# commit
blog-## ;
COMMIT
blog=#

blog=# select * from table_transactions;
 id |  text  | num
-----+-----
  8 | Text2  |  2
 10 | some tex |  2
(2 рядки)

blog=# delete from table_transactions where id = 8;
ERROR:  could not serialize access due to concurrent delete
blog-!# commit
blog-!# ;
ROLLBACK
blog=#
```

На даному рівні ізоляції ми можемо отримати максимальну узгодженість даних і можемо бути впевнені, що зайві дані не будуть зафіксовані.