



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Лабораторна робота № 3
з дисципліни “Бази даних”

Виконав

Студент 2

курсу групи

КП-03

Шалак Данило Володимирович
(прізвище, ім'я, по батькові)

варіант 22

Перевірів

“__” “_____” 20_р.

викладач

Радченко Костянтин
Олександрович
(прізвище, ім'я, по батькові)

Київ 2021

Мета

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

Завдання

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

Умови відповідно варіанту:

Види індексів	Умови для тригера
Hash, BRIN	before delete, insert

Хід роботи

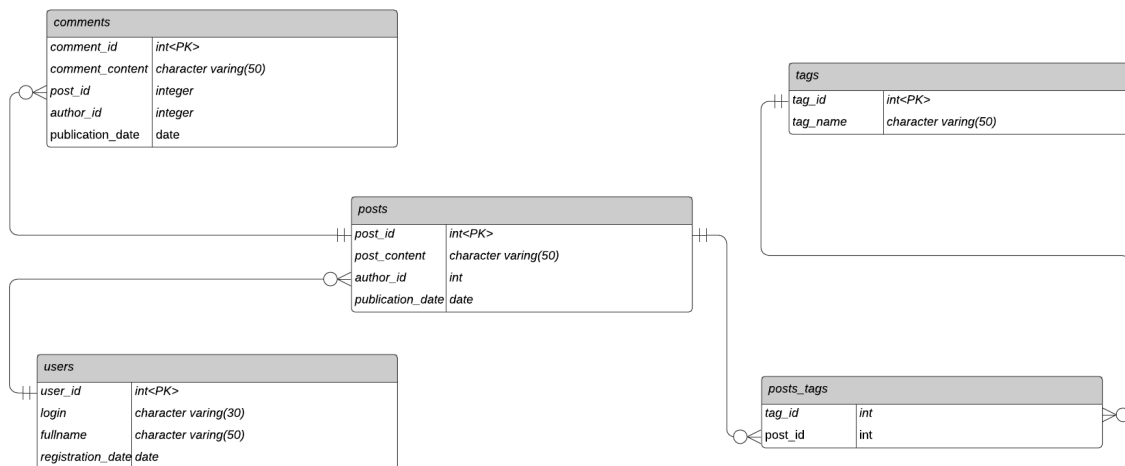


Схема бази даних

Класи таблиць бази даних

```
from sqlalchemy.orm import declarative_base
from sqlalchemy import Column, Integer, String, DateTime, Sequence, ForeignKey

Base = declarative_base()

class Comment(Base):
    __tablename__ = 'comments'

    comment_id = Column(Integer, Sequence('comments_comment_id_seq'),
primary_key=True)
    comment_content = Column(String(100), nullable=False)
    post_id = Column(Integer, ForeignKey('posts.post_id',
ondelete="CASCADE"), nullable=False)
    author_id = Column(Integer, ForeignKey('users.user_id',
ondelete="CASCADE"), nullable=False)
    publication_date = Column(DateTime, nullable=False)

    def __repr__(self):
        return "<Comment(comment_content='%s', post_id='%s', author_id='%s',
publication_date='%s')>" % (
            self.comment_content, self.post_id, self.author_id,
self.publication_date)

class User(Base):
    __tablename__ = 'users'

    user_id = Column(Integer, Sequence('users_user_id_seq'), primary_key=True)
    login = Column(String(30), nullable=False)
    fullname = Column(String(50), nullable=False)
    registration_date = Column(DateTime, nullable=False)

    def __repr__(self):
        return "<User(login='%s', fullname='%s', registration_date='%s')>" % (
            self.login, self.fullname, self.registration_date)

class Post(Base):
    __tablename__ = 'posts'

    post_id = Column(Integer, Sequence('posts_post_id_seq'), primary_key=True)
    post_content = Column(String(100), nullable=False)
    author_id = Column(Integer, ForeignKey('users.user_id',
ondelete="CASCADE"), nullable=False)
    publication_date = Column(DateTime, nullable=False)
```

```

def __repr__(self):
    return "<Post(post_content='%s', author_id='%s', publication_date='%s')>"
% (
        self.post_content, self.author_id, self.publication_date)

class Tag(Base):
    __tablename__ = 'tags'

    tag_id = Column(Integer, Sequence('tags_tag_id_seq'), primary_key=True)
    tag_name = Column(String(50), nullable=False)

    def __repr__(self):
        return "<Tag(tag_name='%s')>" % (self.tag_name)

class PostsTags(Base):
    __tablename__ = 'posts_tags'

    post_id = Column(Integer, ForeignKey('posts.post_id', ondelete='CASCADE'),
primary_key=True)
    tag_id = Column(Integer, ForeignKey('tags.tag_id', ondelete='CASCADE'),
primary_key=True)

    def __repr__(self):
        return "<PostsTags(post_id='%s', tag_id='%s')>" % (self.post_id,
self.tag_id)

```

Приклади запитів у вигляді ORM (видалення, створення, редагування коментарів в класі CommentRepository)

```

def insertComment(self, commentData):
    newComment = Comment(comment_content=commentData[0],
        post_id=commentData[1], author_id=commentData[2],
        publication_date = commentData[3])
    self.session.add(newComment)
    self.session.commit()

def updateComment(self, commentData):
    update_comment = self.session.query(Comment).get(commentData[1])
    update_comment.comment_content = commentData[1]
    self.session.commit()

def deleteComment(self, commentData):
    self.session.delete(self.session.query(Comment).get(commentData[0]))
    self.session.commit()

```



Команди додавання індексів

```
create index name_hash on users using hash(fullname);  
create index reg_date_index on users using brin(registration_date);  
create index comment_author_index on comments using hash (author_id);  
create index post_author_index on posts using hash (author_id);
```



Додавання hash індекса до поля fullname таблиці users (Виконаний

SQL-запит: `explain analyze select * from users where fullname = 'MX';`

До використання індекса

Data Output	Explain	Messages	Notifications
	QUERY PLAN text		
1	Seq Scan on users (cost=0.00..1985.00 rows=157 width=25) (actual time=0.041..12.965 rows=153 loops=1)		
2	[...] Filter: ((fullname)::text = 'MX'::text)		
3	[...] Rows Removed by Filter: 99847		
4	Planning time: 0.115 ms		
5	Execution time: 12.995 ms		

Після використання індекса

	QUERY PLAN text		
1	Bitmap Heap Scan on users (cost=5.22..387.93 rows=157 width=25) (actual time=0.075..0.248 rows=153 loops=1)		
2	[...] Recheck Cond: ((fullname)::text = 'MX'::text)		
3	[...] Heap Blocks: exact=139		
4	[...] -> Bitmap Index Scan on fullname_index (cost=0.00..5.18 rows=157 width=0) (actual time=0.045..0.045 rows=153 loops=1)		
5	[...] Index Cond: ((fullname)::text = 'MX'::text)		
6	Planning time: 0.170 ms		
7	Execution time: 0.295 ms		

Результат виконання цієї команди в консольному додатку (виклик команди `findUser {fullname}`, якщо fullname складається з двох слів, тоді замість пробілу використовувати `'_'`):

```

Enter command
findUser MX
user_id      login      fullname      registration_date
108          user_108    MX            2021-11-08 16:28:07.122748
2973         user_2973    MX            2021-11-10 16:13:07.122748
3268         user_3268    MX            2021-11-10 21:08:07.122748
4553         user_4553    MX            2021-11-11 18:33:07.122748
5984         user_5984    MX            2021-11-12 18:24:07.122748
6416         user_6416    MX            2021-11-13 01:36:07.122748
6645         user_6645    MX            2021-11-13 05:25:07.122748
6999         user_6999    MX            2021-11-13 11:19:07.122748
7990         user_7990    MX            2021-11-14 03:50:07.122748

```

В даному випадку ми отримали значне пришвидшення роботи запиту, це є завдяки тому, що ми перебираємо не всі записи, а тільки необхідні.

Додано brin index до поля registration_date. Виконаємо запит де знайдемо запит де знайде користувачів, які зареєструвались в певний період часу (Виконаний SQL-запит: `select * from users where registration_date between '2021-11-08 14:42:07.122748' and '2021-11-08 14:50:07.122748'`)

До використання індексів

Data Output	Explain	Messages	Notifications
<div> <div>QUERY PLAN</div> <div>text</div> <div></div> </div>			
1	Seq Scan on users (cost=0.00..2235.00 rows=1 width=25) (actual time=0.018..15.900 rows=9 loops=1)		
2	[...] Filter: ((registration_date >= '2021-11-08 14:42:07.122748'::timestamp without time zone) AND (registrati...		
3	[...] Rows Removed by Filter: 99991		
4	Planning time: 0.118 ms		
5	Execution time: 15.932 ms		

Після використання індексів

1	Bitmap Heap Scan on users (cost=12.03..997.04 rows=1 width=25) (actual time=0.043..2.643 rows=9 loops...		
2	[...] Recheck Cond: ((registration_date >= '2021-11-08 14:42:07.122748'::timestamp without time zone) AND (...		
3	[...] Rows Removed by Index Recheck: 17532		
4	[...] Heap Blocks: lossy=128		
5	[...] -> Bitmap Index Scan on reg_date_index (cost=0.00..12.03 rows=16667 width=0) (actual time=0.036..0.0...		
6	[...] Index Cond: ((registration_date >= '2021-11-08 14:42:07.122748'::timestamp without time zone) AND (regi...		
7	Planning time: 1.174 ms		
8	Execution time: 2.696 ms		

Результат виконання цієї команди в консольному додатку (виклик команди

periodUser {startDate} {endDate} якщо вводити дату та час, тоді відокремлювати дату від часу використовуючи '/')

```
Enter command
periodUser 2021-11-08/14:42:07.122748 2021-11-08/14:50:07.122748
```

user_id	login	fullname	registration_date
2	user_2	KV	2021-11-08 14:42:07.122748
3	user_3	NT	2021-11-08 14:43:07.122748
4	user_4	GG	2021-11-08 14:44:07.122748
5	user_5	SJ	2021-11-08 14:45:07.122748
6	user_6	CN	2021-11-08 14:46:07.122748
7	user_7	HA	2021-11-08 14:47:07.122748
8	user_8	CR	2021-11-08 14:48:07.122748
9	user_9	TB	2021-11-08 14:49:07.122748
10	user_10	BH	2021-11-08 14:50:07.122748

В даному випадку також отримали підвищення швидкості виконання запиту за рахунок того, що за допомогою bpin індекса ти розділили таблицю на зони і ми перевіряємо тільки ті зони, які можуть містити шукані значення.

Додано hash індекс до поля author_id таблиці comments та виконано запит для пошуку коментарів користувача за id `select * from comments where author_id = 1`

До використання індекса	
Data Output	Explain Messages Notifications
 QUERY PLAN	
1	Seq Scan on comments (cost=0.00..3963.00 rows=2114 width=23) (actual time=0.017..18.398 rows=2100 loops=1)
2	[...] Filter: (author_id = 1)
3	[...] Rows Removed by Filter: 207900
4	Planning time: 0.097 ms
5	Execution time: 18.491 ms
Після використання індекса	
 QUERY PLAN	
1	Bitmap Heap Scan on comments (cost=68.38..1489.93 rows=2114 width=23) (actual time=0.244..0.722 rows=2100 loops=1)
2	[...] Recheck Cond: (author_id = 1)
3	[...] Heap Blocks: exact=17
4	[...] -> Bitmap Index Scan on comment_author_index (cost=0.00..67.86 rows=2114 width=0) (actual time=0.228..0.228 rows=2100 loops=1)
5	[...] Index Cond: (author_id = 1)
6	Planning time: 0.124 ms
7	Execution time: 0.860 ms

Результат виконання цієї команди в консольному додатку (виклик команди `commentsUser {user_id}`):

```
Enter command
commentsUser 2
comment_id      comment_content  post_id  author_id  pub_date
110101          EY              5         2          2021-11-08 19:57:18.177803
110102          YQ              5         2          2021-11-08 19:58:18.177803
110103          DS              5         2          2021-11-08 19:59:18.177803
110104          HP              5         2          2021-11-08 20:00:18.177803
110105          VS              5         2          2021-11-08 20:01:18.177803
110106          KO              5         2          2021-11-08 20:02:18.177803
110107          HP              5         2          2021-11-08 20:03:18.177803
110108          GD              5         2          2021-11-08 20:04:18.177803
110109          MM              5         2          2021-11-08 20:05:18.177803
110110          JN              5         2          2021-11-08 20:06:18.177803
```

Отримали пришвидшення виконання операції за рахунок зменшення кількості перевірюваних записів в таблиці.

Додаємо hash індекс до поля `author_id` таблиці `posts` та виконаємо запит на знаходження кількості коментарів та постів користувача. (Виконаний SQL-запит:)

```
with comm (comments_count) as
(
    select count(*) from comments where author_id = 2
),
post (posts_count) as
(
    select count (*) from posts where author_id = 2
)
select * from comm, post
```

До використання індекса

	<div>QUERY PLAN</div> <div>text</div> <div></div>
1	Nested Loop (cost=4044.09..4044.14 rows=1 width=16) (actual time=55.647..59.163 rows=1 loops=1)
2	[...] CTE comm
3	[...] -> Finalize Aggregate (cost=3885.34..3885.35 rows=1 width=8) (actual time=54.588..58.096 rows=1 loops=1)
4	[...] -> Gather (cost=3885.23..3885.34 rows=1 width=8) (actual time=29.773..58.085 rows=2 loops=1)
5	[...] Workers Planned: 1
6	[...] Workers Launched: 1
7	[...] -> Partial Aggregate (cost=2885.23..2885.24 rows=1 width=8) (actual time=14.597..14.597 rows=1 loops=2)
8	[...] -> Parallel Seq Scan on comments (cost=0.00..2882.12 rows=1244 width=0) (actual time=0.009..14.454 rows=1050 lo...
9	[...] Filter: (author_id = 1)
10	[...] Rows Removed by Filter: 103950
11	[...] CTE post
12	[...] -> Aggregate (cost=158.73..158.74 rows=1 width=8) (actual time=1.048..1.049 rows=1 loops=1)
13	[...] -> Seq Scan on posts (cost=0.00..158.73 rows=1 width=0) (actual time=0.051..1.043 rows=1 loops=1)
14	[...] Filter: (author_id = 1)
15	[...] Rows Removed by Filter: 8057
16	[...] -> CTE Scan on comm (cost=0.00..0.02 rows=1 width=8) (actual time=54.591..54.592 rows=1 loops=1)
17	[...] -> CTE Scan on post (cost=0.00..0.02 rows=1 width=8) (actual time=1.053..1.054 rows=1 loops=1)
18	Planning time: 0.258 ms
19	Execution time: 59.261 ms

Після використання індекса

	<div>QUERY PLAN</div> <div>text</div> <div></div>
1	Nested Loop (cost=1503.26..1503.31 rows=1 width=16) (actual time=1.238..1.241 rows=1 loops=1)
2	[...] CTE comm
3	[...] -> Aggregate (cost=1495.22..1495.23 rows=1 width=8) (actual time=1.214..1.215 rows=1 loops=1)
4	[...] -> Bitmap Heap Scan on comments (cost=68.38..1489.93 rows=2114 width=0) (actual time=0.255..0.897 rows=2100 l...
5	[...] Recheck Cond: (author_id = 1)
6	[...] Heap Blocks: exact=17
7	[...] -> Bitmap Index Scan on comm_author_ind (cost=0.00..67.86 rows=2114 width=0) (actual time=0.236..0.237 rows=21...
8	[...] Index Cond: (author_id = 1)
9	[...] CTE post
10	[...] -> Aggregate (cost=8.02..8.03 rows=1 width=8) (actual time=0.016..0.017 rows=1 loops=1)
11	[...] -> Index Scan using post_author_ind on posts (cost=0.00..8.02 rows=1 width=0) (actual time=0.013..0.015 rows=1 loo...
12	[...] Index Cond: (author_id = 1)
13	[...] -> CTE Scan on comm (cost=0.00..0.02 rows=1 width=8) (actual time=1.218..1.218 rows=1 loops=1)
14	[...] -> CTE Scan on post (cost=0.00..0.02 rows=1 width=8) (actual time=0.017..0.018 rows=1 loops=1)
15	Planning time: 2.491 ms
16	Execution time: 1.401 ms

Результат виконання цієї команди в консольному додатку (виклик команди `userData {user_id}`):

```
Enter command
userData 2
comments_count      posts_count
2100                1
```

До того як додали індекс до поля `author_id` таблиці `posts` запит виконується помаліше, незважаючи на те, що індекс доданий до поля `author_id` таблиці `comments`. Записи в таблиці `comments` шукаються швидко, але в таблиці `posts` повільно, тому після додавання індекса в таблиці `posts` два запити виконуються швидше.

Тригер `delete_post_trigger` додає записи до таблиці `journal` з деталями про операції і сутності, над якими були виконані певні операції. При додаванні поста додає інформацію про його в цю таблицю, при видаленні додає запис про пост, який видалили та пов'язані з ним коментарі.

Код створення тригера:

```
create function delete_post_trigger() returns trigger
as $delete_post_trigger$
declare comment record;
begin
    if (TG_OP = 'DELETE') then
        for comment in select * from comments where post_id = old.post_id
        loop
            insert into journal(procedure_name, entity_id, entity_type,
entity_content, author_id, post_id, procedure_time)
            values ('DELETE', comment.comment_id, 'comment', comment.comment_content,
comment.author_id, comment.post_id, now());
        end loop;
        insert into journal(procedure_name, entity_id, entity_type, entity_content,
author_id, procedure_time)
        values ('DELETE', old.post_id, 'post', old.post_content, old.author_id, now());
        return old;
    elsif (TG_OP = 'INSERT') then
        if (length(new.post_content) > 99) then
            raise exception 'post length must be less then 100 characters';
        end if;

        insert into journal(procedure_name, entity_id, entity_type, entity_content,
author_id, procedure_time)
        values ('INSERT', new.post_id, 'post', new.post_content, new.author_id, now());
        return new;
    end if;
```

```

end;
$delete_post_trigger$ language plpgsql;

CREATE TRIGGER delete_post_trigger
before DELETE or INSERT ON posts
FOR EACH ROW EXECUTE PROCEDURE delete_post_trigger();

```

Приклади роботи тригера в різних ситуаціях:

Коли намагаємось додати пост до таблиці бази даних і кількість символів перевищує допустимі значення (команда insertPost консольного додатку) :










Data Output Explain Messages Notifications

```

ERROR:  post length must be less then 100 characters
CONTEXT:  PL/pgSQL function delete_post_trigger() line 16 at RAISE
SQL state: P0001

```

Записи в таблиці journal після видалення поста(команда deletePost консольного додатку):

Data Output									Explain	Messages	Notifications
	 id [PK] integer 	procedure_name character varying (20) 	entity_id integer 	entity_type character varying (20) 	entity_content character varying (100) 	author_id integer 	post_id integer 	procedure_time timestamp without time zone 			
17	17	INSERT	101021	post	Hello world	234	[null]	2021-11-10 18:19:50.819373			
18	18	DELETE	331023	comment	skdfbsdf	564	101021	2021-11-10 18:20:49.607495			
19	19	DELETE	331024	comment	uwefkdv	564	101021	2021-11-10 18:20:49.607495			
20	20	DELETE	331025	comment	jijlojsadc	564	101021	2021-11-10 18:20:49.607495			
21	21	DELETE	331026	comment	lkdfjidhfvd	564	101021	2021-11-10 18:20:49.607495			
22	22	DELETE	101021	post	Hello world	234	[null]	2021-11-10 18:20:49.607495			

Висновки

В ході лабораторної роботи я навчився використовувати sqlalchemy для виконання операцій з сутностями в базі даних. Навчився створювати індекси до полів таблиці, проаналізував, коли слід використовувати певний тип індексів. Навчився створювати тригери до при виконанні певних операцій над сутностями таблиці.