



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 II курсу

групи КП-03

Заїка Максим Олександрович

Перевірив

“ ____ ” “ _____ ” 20__ р.

викладач

Радченко Костянтин

Олександрович

варіант № 4

Київ 2021

Тема

Засоби оптимізації роботи СУБД PostgreSQL

Мета

Здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

Завдання

11. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи No2

у вигляд об’єктно-реляційної проєкції (ORM).

2. Створити та проаналізувати різні типи індексів у PostgreSQL.

3. Розробити тригер бази даних PostgreSQL.

Хід роботи

Класи сутностей у вигляді ORM:

```
class Author(Base):
    __tablename__ = 'authors'
    id = Column(Integer, primary_key=True)
    name = Column(String)
    birthday = Column(Date)
    books = relationship('Book', cascade='all, delete')

    def __repr__(self):
        return '<Author(id={}, name={}, birthday={})>'.\
            format(self.id, self.name, self.birthday)
```

```
class Book(Base):
    __tablename__ = 'books'
    id = Column(Integer, primary_key=True)
    year = Column(Integer)
    pages = Column(Integer)
    author_id = Column(Integer, ForeignKey('authors.id'))
    name = Column(String)

    def __repr__(self):
        return '<Book(id={}, year={}, pages={}, author_id={}, name={})>'.\
            format(self.id, self.year, self.pages, self.author_id, self.name)
```

```
class Reader(Base):
    __tablename__ = 'readers'
    id = Column(Integer, primary_key=True)
    money = Column(Integer)
    birthday = Column(Date)
    pass_id = Column(Integer, ForeignKey('passes.id'))

    def __repr__(self):
        return '<Reader(id={}, money={}, birthday={}, pass_id={})>'.\
            format(self.id, self.money, self.birthday, self.pass_id)
```

Приклади запитів:

```
r = Repository()
book = Book(year=1000, pages=300, author_id=5, name='the best book ever')
r.insert(book)
r.get(entity=Book, condition=f"id = {book.ID}")
r.update(entity=Book, condition=f"id = {book.ID}", values="pages = 400")
r.delete(entity=Book, condition=f"id = {book.ID}")
```

Виконання запитів до створення індексів:

```
5 select * from authors where name like '%A%' group by id order by birthday desc
```

6

Data Output	Explain	Messages	Notifications
-------------	---------	----------	---------------

Successfully run. Total query runtime: 173 msec.
33351 rows affected.

```
6 select * from authors where id > 10 group by id order by birthday desc
```

Data Output	Explain	Messages	Notifications
-------------	---------	----------	---------------

Successfully run. Total query runtime: 266 msec.
100008 rows affected.

Команди створення індексів:

```
1 create index gin_on_author_name on books using gin(to_tsvector('english', name));
2 create index brin_on_author_id on books using brin(id);
```

Виконання запитів після створення індексів:

```
4 select * from authors where name like '%A%' group by id order by birthday desc
```

Data Output	Explain	Messages	Notifications
-------------	---------	----------	---------------

Successfully run. Total query runtime: 121 msec.
33351 rows affected.

```
5 select * from authors where id > 10 group by id order by birthday desc
```

Data Output	Explain	Messages	Notifications
-------------	---------	----------	---------------

Successfully run. Total query runtime: 189 msec.
100008 rows affected.

Індекси прискорюють виконання запитів, оскільки GIN-індекси призначені для оптимізації пошуку на тексту шляхом розбиття тексту на атомарні

складові і пошук за цими складовими; BRIN індекси прискорюють пошук, оскільки id книг розподілений рівномірно в базі даних, тому можна легко і з користю відсортувати, розбити таблицю id на менші сегменти.

Для демонстрації роботи тригера було створено додаткову таблицю, в яку запитується ім'я таблиці та ім'я івента, пов'язаних зі змінами. До того ж, для цієї таблиці був створений before insert trigger для автоматичної генерації часу про зміни.

```
1 create table if not exists infolog
2 (
3     tablename text,
4     eventname text,
5     time timestamp
6 );
7 create or replace function on_info_insert() returns trigger as $on_info_user$
8 begin
9     new.time := clock_timestamp();
10    return new;
11 end;
12 $$ language plpgsql;
13
14 drop trigger if exists on_info_insert on infolog;
15 create trigger on_info_insert before insert on infolog
16     for each row execute procedure on_info_insert();
```

В таблиці books було створено 2 after delete, after insert тригера:

```
1 create or replace function on_insert_book() returns trigger as
2 $$
3 begin
4     if new.year < 0 then
5         raise exception 'Year can't be less than 0. Given %', new.year;
6     end if;
7     if new.pages < 0 then
8         raise exception 'Number of pages can't be less than 0. Given %', new.pages;
9     end if;
10    insert into infolog (tablename, eventname) values ('books', 'insert');
11    return new;
12 end;
13 $$ language plpgsql;
14
15 drop trigger if exists on_insert_book on books;
16 create trigger on_insert_book after insert on books
17     for each row execute procedure on_insert_book();
```

```

1 create or replace function on_delete_book() returns trigger as
2 $$
3 ▼ begin
4     insert into infolog (tablename, eventname) values ('books', 'delete');
5     return new;
6 end;
7 $$ language plpgsql;
8
9 drop trigger if exists on_delete_book on books;
10 create trigger on_delete_book after delete on books
11     for each row execute procedure on_delete_book();

```

Перевірка виняткових ситуацій:

```

1 insert into books (year, pages, author_id, name) values (1900, -200, 3, 'Some book')

```

Data Output Explain Messages Notifications

ERROR: ПОМИЛКА: Number of pages can't be less than 0. Given -200
 CONTEXT: Функція PL/pgSQL on_insert_book() рядок 7 в RAISE

SQL state: P0001

```

1 insert into books (year, pages, author_id, name) values (-1, 200, 3, 'Some book')

```

Data Output Explain Messages Notifications

ERROR: ПОМИЛКА: Year can't be less than 0. Given -1
 CONTEXT: Функція PL/pgSQL on_insert_book() рядок 4 в RAISE




Приклад справної роботи всіх тригерів:

```

1 insert into books (year, pages, author_id, name) values (1000, 200, 3, 'Some book');
2 select * from infolog;

```

Data Output Explain Messages Notifications

	 tablename text	 eventname text	 time timestamp without time zone
1	books	insert	2021-11-27 15:19:50.150682

```
1 delete from books where name = 'Some book';
2 select * from infolog;
```

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

	tablename text	eventname text	time timestamp without time zone
1	books	insert	2021-11-27 15:19:50.150682
2	books	delete	2021-11-27 15:22:10.088234

Висновки

У даній роботі було вивчено ORM представлення інструментів БД за допомогою бібліотеки sqlalchemy. Вивчено 4 види індексів, їх переваги та недоліки, використано на практиці 2 типи з них, проаналізовано їх ефективність. Створено 3 тригера для однієї таблиці.