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**»**

Виконав студент III курсу
ФПМ групи КВ-83
Панфілов Я.Ю.
Перевірив: Павловський В.І.

Київ – 2020

Засоби оптимізації роботи СУБД PostgreSQL

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.

Варіант 18

У другому завданні проаналізувати індекси Btree, Gin.
Умова для тригера – after insert, update

Завдання 1

Логічна схема бази даних “Блог”

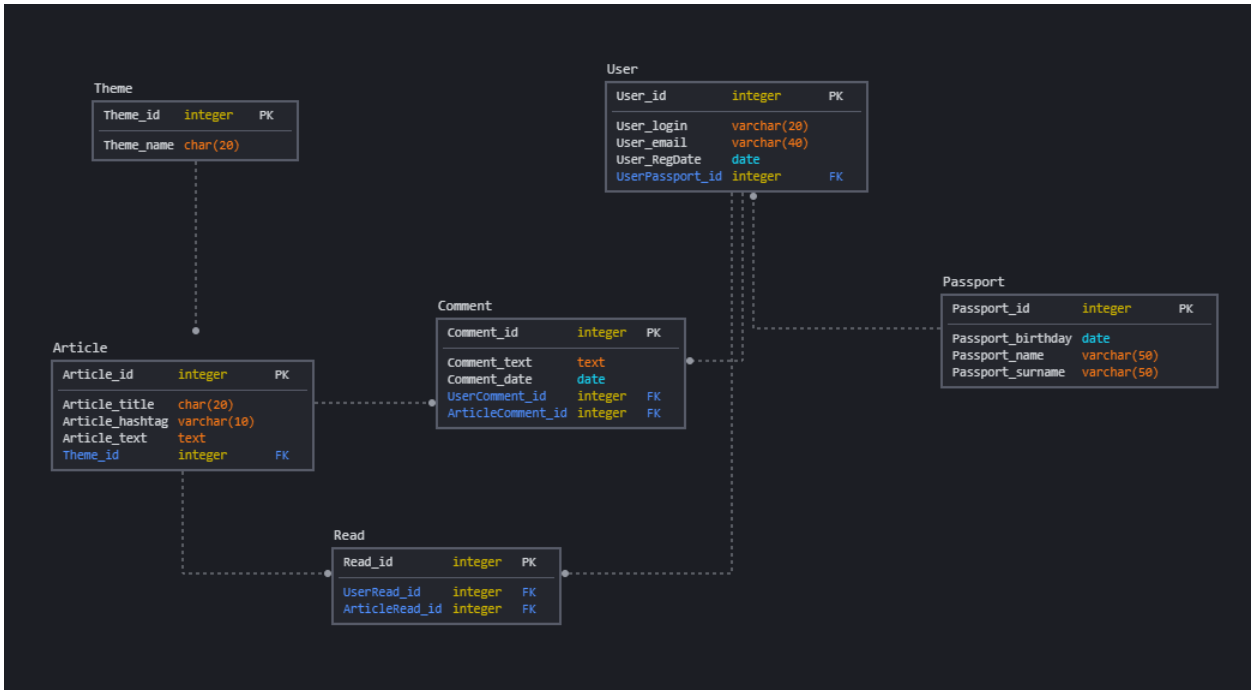


Рис 3.1 – Логічна модель даних

Для перетворення модулю “Model” програми, створеної в 2 лабораторній роботі, у вигляд об’єктно-реляційної моделі використовую бібліотеку SqlAlchemy, яка є найпопулярнішою на мові Python.

Зобразимо сутнісні класи програми. Продемонструємо код для класу Article:

```
class Article(Base):
    __tablename__ = 'article'
    id = Column('article_id',Integer, primary_key=True)
    article_title = Column(String(50))
    article_hashtag = Column(DateTime(timezone=False))
    article_text = Column(Integer)
    article_theme = Column(Integer,ForeignKey('theme.theme_id'))

    def __init__(self,data):
        self.article_title = data['article_title']
        self.article_hashtag = data['article_hashtag']
        self.article_text = data['article_text']
        self.article_theme = int(data['article_theme'])

    def change_data(self,data):

        for key,value in data.items():

            if key == 'article_hashtag':
                self.article_hashtag = value
            elif key == 'article_text':
                self.article_text = value
            elif key == 'article_theme':
                self.article_theme = int(value)
            else:
                setattr(self,key,value)
    def to_cortege(self):
        return (self.id,self.article_title,self.article_hashtag,self.article_text,self.article_theme)

    def get_column_names():
        return ['article_id','article_title','article_hashtag','article_text','article_theme']
```

В класі Music, описується вміст таблиці і зв’язки з іншими таблицями

В цій ORM уже є готові базові CRUD (create, read, update, delete) методи. Метод для створення запису виглядає так:

```
def insert_data(self,table_name,values):

    table = self.get_type_by_name(table_name)
    obj = table(values)
    self.__session.add(obj)
    self.__session.commit()
```

Метод для оновлення даних:

```
def change_data(self,table_name,values):

    cond = values.pop('condition')
    table = self.get_type_by_name(table_name)
    q = self.__session.query(table).filter_by(id = cond).first()
    q.change_data(values)
    self.__session.flush()
    self.__session.commit()
```

Метод для видалення даних:

```
def delete_data(self,table_name,cond):
    data = None
    table = self.get_type_by_name(table_name)
    try:
        data = self.__session.query(table).all()
    except Exception as e:
        return str(e)

    return ( table.get_column_names() , [row.to_cortege() for row in data])
```

Завдання 2

Btree

Для дослідження індексу була створена таблиця test_btree, яка має дві колонки: числову і текстову. Вони проіндексовані як BTree. У таблицю було занесено 1000000 записів.

Виконуємо запити для пошуку:

```
select count(*) as num_count from "test_btree" where "num_col" % 2 = 0;
```

```
select count(*) as num_count from "test_btree" where "num_col" % 2 = 0 or "text_col" like '%rty%';
```

Результат:

	num_count bigint
1	500440

	num_count bigint
1	534126

У неіндексованій таблиці:

Successfully run. Total query runtime: 165 msec. 1 rows affected.

Successfully run. Total query runtime: 145 msec. 1 rows affected.

У проіндексованій таблиці:

Successfully run. Total query runtime: 125 msec. 1 rows affected.

Successfully run. Total query runtime: 133 msec. 1 rows affected.

GIN

Для дослідження індексу була створена таблиця test_gin, яка має колонку типу text та колонку типу tsvector. У таблицю було занесено 1000000 записів.

Знайдемо слова, які зустрічаються найрідше:

```
select word, ndoc from ts_stat('select tsvec from "test_gin"') order by ndoc asc limit 5
```

Результат:

	word text	ndoc integer
1	lzxcvbnmqw	18935
2	vbnmqwerti	19120
3	xcvbnm	19122
4	nmqwertyui	19153
5	lzxcvbnm	19233

Здійснимо пошук по цих словах:

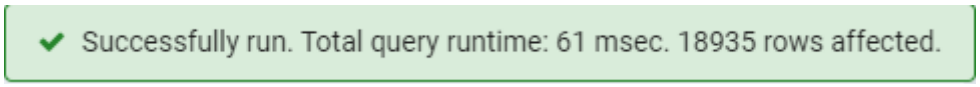
```
select * from test_gin where tsvec @@ to_tsquery('lzxcvbnmqw')
```

Результат:

	text_col text	tsvec tsvector
1	lzxcvbnmQW	'lzxcvbnmq...
2	lzxcvbnmQW	'lzxcvbnmq...
3	lzxcvbnmQW	'lzxcvbnmq...
4	lzxcvbnmQW	'lzxcvbnmq...
5	lzxcvbnmQW	'lzxcvbnmq...
6	lzxcvbnmQW	'lzxcvbnmq...
7	lzxcvbnmQW	'lzxcvbnmq...
8	lzxcvbnmQW	'lzxcvbnmq...
9	lzxcvbnmQW	'lzxcvbnmq...
10	lzxcvbnmQW	'lzxcvbnmq...
11	lzxcvbnmQW	'lzxcvbnmq...
12	lzxcvbnmQW	'lzxcvbnmq...

Час виконання для неіндексованої таблиці:

Час виконання для проіндексованої таблиці:



Завдання 3

Тригер:

```
create or replace function update_insert_func() returns trigger as $$

declare
    cursor_log cursor for select * from "buildLog";
    row_log "buildLog"%rowtype;

begin
    if new."build_name" like '%newbuild%' then
        insert into "buildLog" ("bName") values (new."build_name");
        return new;
    else
        raise notice 'Not newbuild!';
        for row_log in cursor_log loop
            update "buildLog" set "bName" = row_log."bName" || 'not' where current of cursor_log;
        end loop;
        return new;
    end if;
end;

$$ language plpgsql;

create trigger test_trigger
after update or insert on "build"
for each row
execute procedure update_insert_func()
```

Принцип роботи:

Тригер спрацьовує при оновленні чи вставці у таблиці build. Якщо значення build_name, яке оновлюється чи вставляється, дорівнює newbuild, то цей новий запис заноситься у таблицю логів buildLog. Якщо значення build_name не дорівнює newbuild, то до кожного значення bName у таблиці buildLog додається “not”.

Вставляємо запис у build:

```
insert into "build" ("build_name") values ('newbuild')
```

buildLog:

	bId [PK] integer	bName text
1	1	newbuild

Оновимо запис:

```
1  update "build" set "build_name" = 'not new build' where "build_name" = 'newbuild'
```

Data Output | Explain | Messages | Notifications

```
ЗАМЕЧАНИЕ:  Not newbuild!
UPDATE 1
```

buildLog:

	bId [PK] integer	bName text
1	1	newbuildnot