



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

## **Лабораторна робота №2**

з дисципліни

**«Бази даних і засоби управління»**

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

Виконав: студент III курсу

ФПМ групи КВ-13

Петраш Антон Степанович

Київ – 2023

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

Завдання роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC РГР 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.

### Варіант 14

Індекси для аналізу у завдання №2: *Btree, Hash*

#### Концептуальна модель предметної області “Система обліку виконавців та виступів на фестивалях”

В концептуальній моделі “Система обліку виконавців та виступів на фестивалях” є наступні сутності та зв’язки:

1. Сутність “Performer” з атрибутами “Artist\_ID”, “name”, “surname”, “genre”
2. Сутність “Performance” з атрибутами “Performance\_ID”, “Festival\_ID”, “Artist\_ID”, “Start\_time”, “Finish\_time”
3. Сутність “Festival” з атрибутами “Festival\_ID”, “Fest\_name”, “Price”, “City”

Зв’язок між виконавцем і виступом - один до багатьох, тому що один виконавець може проводити багато виступів, але один виступ проводить один виконавець. Зв’язок між Фестивалем і виступом - один до багатьох, адже на одному фестивалі може бути багато виступів і кожен виступ належить одному фестивалю.

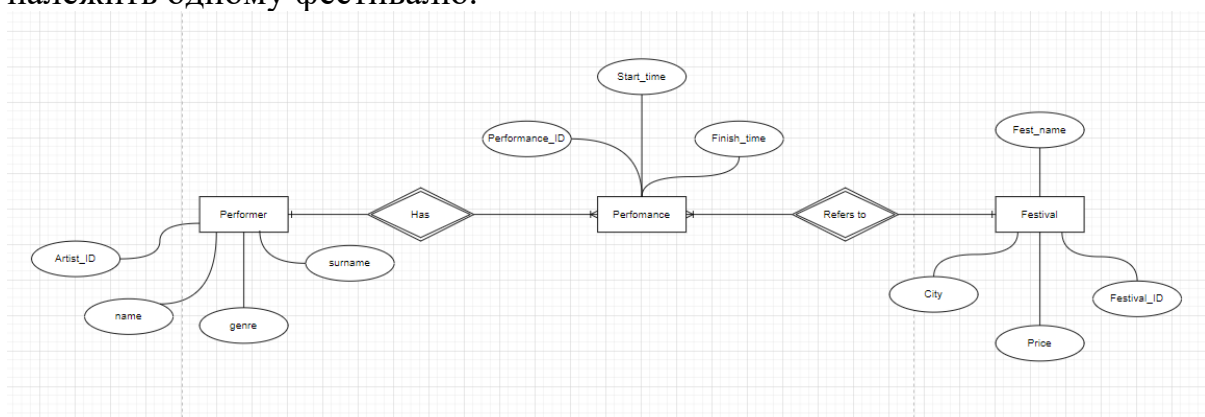


Рис.1 концептуальна модель “Система обліку виконавців та виступів на фестивалях”

#### Логічна модель “Система обліку виконавців та виступів на фестивалях”

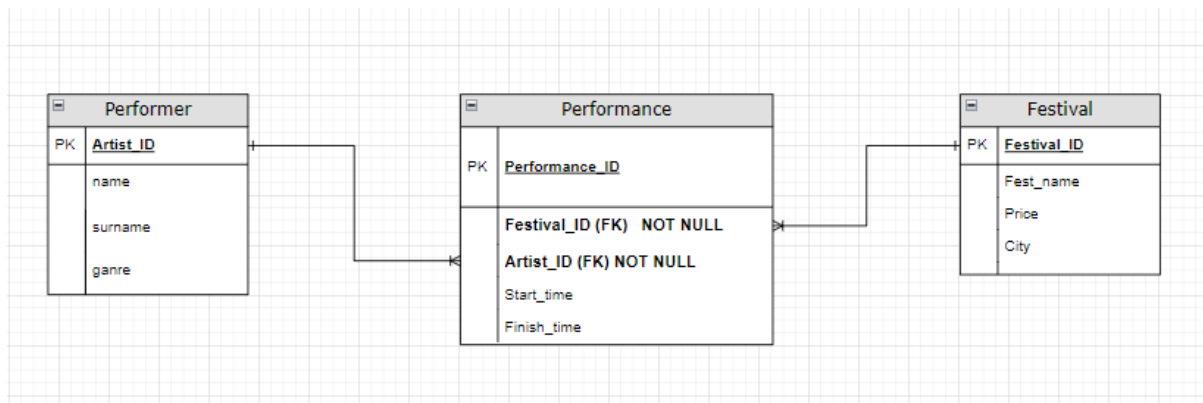


Рис.1 логічна модель “Система обліку виконавців та виступів на фестивалях”

Середовище для налаштування, підключення та розробки бази даних

Мова програмування : Python 3.11

Модуль “psycopg2” був використаний для підключення до сервера бази даних

Для перетворення модуля “Model” у вигляд об’єктно-реляційної моделі використовується бібліотека «SQLAlchemy”

## Завдання 1

Для перетворення функцій, що реалізують запити до об’єктної бази даних, необхідно встановити бібліотеку sqlalchemy, налаштувати програму на роботу з ORM, розробити класи-сутності для об’єктів-сутностей, представлених відповідними таблицями БД та пов’язаних зв’язками 1:M, M:M та 1:1 виконати опис схеми бази даних. Особливу увагу приділити контролю зовнішніх зв’язків між таблицями засобами ORM.

Замінити виклики запитів мовою SQL на відповідні запити засобами SQLAlchemy по роботі з об’єктами. Обов’язковим є реалізація вставки, видалення та редагування екземплярів класів-сутностей. Розробка запитів на генерацію даних та пошук екземплярів класів-сутностей вітається, але не є обов’язковою.

Інтерфейси функцій (вхідні та вихідні аргументи функцій модуля “Модель”) мають залишитись без змін.

Оновлений код програми:

Новий модуль “alch” використовується для підключення бібліотеки

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

DATABASE_URL = 'postgresql://postgres:qwerty@localhost:5432/postgres'
engine = create_engine(DATABASE_URL)

Base = declarative_base()

Session = sessionmaker()
```

Оновлення модуля Performer/model:

```
import alch
from sqlalchemy import Column, Integer, String

3 usages  Anton Petrash
class Performer(alch.Base):
    __tablename__ = 'Performer'
    Artist_ID = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)
    surname = Column(String, nullable=False)
    genre = Column(String, nullable=True)

class ModelPerformer:
    Anton Petrash
    def __init__(self, db_model):
        self.conn = db_model.conn
        self.engine = alch.create_engine(alch.DATABASE_URL)
        self.session = alch.Session.configure(bind=self.engine)
        self.session = alch.Session()
```

1 usage (1 dynamic) Anton Petrash

```
def add_Performer(self, Artist_ID, name, surname, genre):
    try:
        new_performer = Performer(
            Artist_ID=Artist_ID,
            name=name,
            surname=surname,
            genre=genre
        )
        self.session.add(new_performer)
        self.session.commit()
        return True # Returns True if the update was successful
    except Exception as e:
        self.conn.rollback()
        print(f"Error With Adding A Performer: {str(e)}")
        return False # Returns False if insertion fails
```

1 usage (1 dynamic) Anton Petrash

```
def update_Performer(self, Artist_ID, name, surname, genre):
    try:
        performer = self.session.query(Performer).filter_by(Artist_ID=Artist_ID).first()

        if performer:
            performer.Artist_ID = Artist_ID
            performer.name = name
            performer.surname = surname
            performer.genre = genre

            self.session.commit()
            return True # Returns True if the update was successful
        else:
            return False
    except Exception as e:
        self.session.rollback()
        print(f"Error With A Performer Updating: {str(e)}")
        return False # Returns False if insertion fails
```

1 usage (1 dynamic) Anton Petrash

```
def delete_Performer(self, Artist_ID):
    try:
        performer = self.session.query(Performer).filter_by(Artist_ID=Artist_ID).first()

        if performer:
            self.session.delete(performer)
            self.session.commit()
            return True # Returns True if the update was successful
        else:
            return False
    except Exception as e:
        self.session.rollback()
        print(f"Error With An Artist Deleting: {str(e)}")
        return False # Returns False if insertion fails
```

Оновлення модуля Festival/model:

```
import alch
from sqlalchemy import Column, Integer, String, Numeric

3 usages  Anton Petrash
class festival(alch.Base):
    __tablename__ = 'Festival'
    Festival_ID = Column(Integer, primary_key=True)
    Fest_name = Column(String, nullable=False)
    Price = Column(Numeric(10, 2), nullable=False)
    City = Column(String, nullable=False)

2 usages  Anton Petrash
class ModelFestival:
    Anton Petrash
    def __init__(self, db_model):
        self.conn = db_model.conn
        self.engine = alch.create_engine(alch.DATABASE_URL)
        self.session = alch.Session.configure(bind=self.engine)
        self.session = alch.Session()

1 usage (1 dynamic)  Anton Petrash
def add_Festival(self, Festival_ID, Fest_name, Price, City):
    try:
        new_festival = festival(
            Festival_ID=Festival_ID,
            Fest_name=Fest_name,
            Price=Price,
            City=City
        )
        self.session.add(new_festival)
        self.session.commit()
        return True # Returns True if the update was successful
    except Exception as e:
        self.session.rollback()
        print(f"Error With Adding A Festival: {str(e)}")
        return False # Returns False if insertion fails
```

1 usage (1 dynamic) Anton Petrash

```
def update_Festival(self, Festival_ID, Fest_name, Price, City):
    try:
        Festival = self.session.query(festival).filter_by(Festival_ID=Festival_ID).first()

        if Festival:
            Festival.Festival_ID = Festival_ID
            Festival.Fest_name = Fest_name
            Festival.Price = Price
            Festival.City = City

            self.session.commit()
            return True # Returns True if the update was successful
        else:
            return False
    except Exception as e:
        self.session.rollback()
        print(f"Error With A Festival Updating: {str(e)}")
        return False # Returns False if insertion fails
```

1 usage (1 dynamic) Anton Petrash

```
def delete_Festival(self, Festival_ID):
    try:
        Festival = self.session.query(festival).filter_by(Festival_ID=Festival_ID).first()

        if Festival:
            self.session.delete(Festival)
            self.session.commit()
            return True # Returns True if the update was successful
        else:
            return False
    except Exception as e:
        self.session.rollback()
        print(f"Error With Deleting A Festival Violates A Foreign Key Constraint: {str(e)}")
        return False # Returns False if insertion fails
```


## Оновлення модуля Performance/model:


```
import alch
from sqlalchemy import Column, Integer, Date
```

2 usages  Anton Petrash

```
class performance(alch.Base):
    __tablename__ = 'Performance'
    Performance_ID = Column(Integer, primary_key=True)
    Festival_ID = Column(Integer, nullable=False)
    Artist_ID = Column(Integer, nullable=False)
    Start_time = Column(Date, nullable=False)
    Finish_time = Column(Date, nullable=False)
```

2 usages  Anton Petrash

```
class ModelPerformance:
     Anton Petrash
    def __init__(self, db_model):
        self.conn = db_model.conn
        self.engine = alch.create_engine(alch.DATABASE_URL)
        self.session = alch.Session.configure(bind=self.engine)
        self.session = alch.Session()
```

1 usage (1 dynamic)  Anton Petrash

```
def add_Performance(self, Performance_ID, Festival_ID, Artist_ID, Start_time, Finish_time):
    try:
        new_performance = performance(
            Performance_ID=Performance_ID,
            Festival_ID=Festival_ID,
            Artist_ID=Artist_ID,
            Start_time=Start_time,
            Finish_time=Finish_time
        )
        self.session.add(new_performance)
        self.session.commit()
        return True # Returns True if the update was successful
    except Exception as e:
        self.session.rollback()
        print(f"Error With Adding A Performance: {str(e)}")
        return False
```



1 usage (1 dynamic) Anton Petrash

```
def update_Performance(self, Performance_ID, Festival_ID, Artist_ID, Start_time, Finish_time):
    try:
        Performance = self.session.query(performance).filter_by(Prefomance_ID=Performance_ID).first()

        if Performance:
            Performance.Performance_ID = Performance_ID
            Performance.Festival_ID = Festival_ID
            Performance.Artist_ID = Artist_ID
            Performance.Start_time = Start_time
            Performance.Finish_time = Finish_time

            self.session.commit()
            return True # Returns True if the update was successful
        else:
            return False
    except Exception as e:
        # Handling an error if the update failed
        self.session.rollback()
        print(f"Error With Updating A Performance: {str(e)}")
        return False # Returns False if insertion fails

def delete_Performance(self, Performance_ID):
    c = self.conn.cursor()
    try:
        # Attempting to update a record
        c.execute('DELETE FROM "Performance" WHERE "Performance_ID"=%s', (Performance_ID,))
        self.conn.commit()
        return True # Returns True if the update was successful
    except Exception as e:
        # Handling an error in case the deletion failed
        self.conn.rollback()
        print(f"Error With Deleting A Performance: {str(e)}")
        return False # Returns False if insertion fails
```

Програма працює так само як і програма з РГР

Main Menu:

1. Add New Performance
2. Show Performances
3. Renewal Performance
4. Remove Performance
5. Add New Festival
6. Show Festivals
7. Renewal Festival
8. Remove Festival
9. Add New Performer
10. Show Performers
11. Renewal Performer
12. Remove Performer
13. Create Data By Random
14. Delete All Data
15. View Analytics
16. Exit

Choose an action :

Choose an action : 5

Input Festival id: 1

Input Festival name: Festival

Input Festival price: 120

Input Festival city: Lviv

Successfully Added A Festival

Choose an action : 7

Input Festival id: 1

Input Festival name: Concert

Input Festival price: 140

Input Festival city: Lviv

Successfully Updated A Festival

Choose an action : 8

Input Festival id: 1

Successfully Deleted A Festival

## Завдання 2

Відповідно до варіанту індексування продемонструвати на прикладах запитів SQL SELECT підвищення швидкодії їх виконання з використанням індексів, а також пояснити чому для деяких випадків індексування використовувати недоцільно. При цьому для наочного представлення слід використати функцію генерування рандомізованих даних з лабораторної роботи №2, створивши необхідну кількість тестових даних. Навести 4-5 прикладів запитів SELECT (із виведенням результуючих даних), що містять фільтрацію, агрегатні функції, групування та сортування (у необхідних комбінаціях).

### Індекс B-tree

Генерація даних:

---

```
CREATE TABLE docs (  
    id serial PRIMARY KEY,  
    title text,  
    content text,  
    keywords varchar(200)[]  
);  
  
INSERT INTO docs (title, content, keywords)  
SELECT  
    'Docs ' || seq,  
    'Sample docs content ' || seq,  
    array[  
        'keyword_' || (seq + floor(random()*30))::int,  
        'keyword_' || (seq + floor(random()*30))::int,  
        'keyword_' || (seq + floor(random()*30))::int  
    ]  
FROM generate_series(1, 1000000) seq;
```

## Запит без індексу

-- Запит, який може не використовувати індекс

```
EXPLAIN ANALYZE SELECT * FROM docs WHERE keywords[1] = 'keyword_5';
```

Output Messages Notifications



### QUERY PLAN

text



Gather (cost=1000.00..25919.33 rows=5000 width=121) (actual time=0.357..203.121 rows=1 loops=1)

Workers Planned: 2

Workers Launched: 2

-> Parallel Seq Scan on docs (cost=0.00..24419.33 rows=2083 width=121) (actual time=63.571..121.421 rows=0 loop...

Filter: ((keywords[1])::text = 'keyword\_5')::text)

Rows Removed by Filter: 333333

Planning Time: 0.373 ms

Execution Time: 203.144 ms

## Запит з індексом

```
2 DROP INDEX IF EXISTS idx_keywords;
```

```
3
```

```
4 -- Запит, який може не використовувати індекс
```

```
5 EXPLAIN ANALYZE SELECT * FROM docs WHERE keywords[1] = 'keyword_5';
```

Data Output Messages Notifications



### QUERY PLAN

text



1 Gather (cost=1000.00..25919.33 rows=5000 width=121) (actual time=0.342..174.332 rows=1 loops=1)

2 Workers Planned: 2

3 Workers Launched: 2

4 -> Parallel Seq Scan on docs (cost=0.00..24419.33 rows=2083 width=121) (actual time=61.676..110.790 rows=0 loop...

5 Filter: ((keywords[1])::text = 'keyword\_5')::text)

6 Rows Removed by Filter: 333333

7 Planning Time: 0.090 ms

8 Execution Time: 174.354 ms

Запит без індексу для меншої кількості даних (пошук за діапазоном)

```
-- Запит, який може не використовувати індекс  
EXPLAIN ANALYZE SELECT * FROM docs WHERE keywords[1] = 'keyword_5';
```

Output Messages Notifications

QUERY PLAN	🔒
text	
Seq Scan on docs (cost=0.00..298.00 rows=50 width=106) (actual time=2.374..2.374 rows=0 loops...	
Filter: ((keywords[1])::text = 'keyword_5'::text)	
Rows Removed by Filter: 10000	
Planning Time: 0.258 ms	
Execution Time: 2.387 ms	

Запит з індексом для меншої кількості даних (пошук за діапазоном)

```
-- Створення індексу B-tree на стовпці keywords  
CREATE INDEX idx_keywords ON docs USING btree(keywords);  
  
-- Запит для використання індексу B-tree  
EXPLAIN ANALYZE SELECT * FROM docs WHERE keywords @> ARRAY['keyword_5']::varchar(200)[];
```

Output Messages Notifications

QUERY PLAN	🔒
text	
Seq Scan on docs (cost=0.00..298.00 rows=3 width=106) (actual time=5.176..5.176 rows=0 loops...	
Filter: (keywords @> '{keyword_5}'::character varying(200)[])	
Rows Removed by Filter: 10000	
Planning Time: 0.380 ms	
Execution Time: 5.192 ms	

Висновок з роботи індексу B-tree

Індекс B-tree призначений для прискорення операцій пошуку та сортування в базі даних. Він добре підходить для роботи з типами даних як можна порівнювати між собою, такі як числа, рядки, дати, тощо.

Але він не ефективний коли є мала кількість записів через необхідність підтримувати структуру дерева.

Це і показано в дослідженні, при великій кількості записів використання індекса значно зменшує час обробки запиту, тоді коли при малій кількості записів використання індекса збільшує час обробки більш ніж вдвічі.

## Індекс *Hash*

Перший запит, пошук точних збігів:

Без індексу:

```
EXPLAIN ANALYZE SELECT * FROM docs WHERE keywords @> ARRAY['keyword_42']::varchar(200)[];
```

Output Messages Notifications



### QUERY PLAN

text

Gather (cost=1000.00..25919.33 rows=5000 width=121) (actual time=0.405..267.872 rows=5 loops=1)

Workers Planned: 2

Workers Launched: 2

-> Parallel Seq Scan on docs (cost=0.00..24419.33 rows=2083 width=121) (actual time=111.669..191.873 rows=2 loop...

Filter: (keywords @> '{keyword\_42}':character varying(200))

Rows Removed by Filter: 333332

Planning Time: 0.107 ms

Execution Time: 267.892 ms

З використанням індексу:

```
EXPLAIN ANALYZE SELECT * FROM docs WHERE keywords @> ARRAY['keyword_42']::varchar(200)[];
```

Output Messages



### QUERY PLAN

text

Gather (cost=1000.00..25919.33 rows=5000 width=121) (actual time=0.314..215.491 rows=5 loops=1)

Workers Planned: 2

Workers Launched: 2

-> Parallel Seq Scan on docs (cost=0.00..24419.33 rows=2083 width=121) (actual time=88.437..150.387 rows=2 loop...

Filter: (keywords @> '{keyword\_42}':character varying(200))

Rows Removed by Filter: 333332

Planning Time: 0.091 ms

Execution Time: 215.506 ms

Використання індексу пришвидшило обробку запиту.

Другий запит, пошук часткових збігів:

Без індексу:

```
EXPLAIN ANALYZE SELECT * FROM docs WHERE title LIKE 'Docs%';
```

Output Messages

QUERY PLAN	lock
text	
Seq Scan on docs (cost=0.00..31711.00 rows=999900 width=121) (actual time=0.056..306.367 rows=1000000 loops...	
Filter: (title ~~ 'Docs%':text)	
Planning Time: 0.287 ms	
Execution Time: 337.465 ms	

З індексом:

```
CREATE INDEX idx_title_hash ON docs USING hash (title);
```

```
EXPLAIN ANALYZE SELECT * FROM docs WHERE title LIKE 'Docs%';
```

Output Messages

QUERY PLAN	lock
text	
Seq Scan on docs (cost=0.00..31711.00 rows=999900 width=121) (actual time=0.041..321.038 rows=1000000 loops...	
Filter: (title ~~ 'Docs%':text)	
Planning Time: 0.100 ms	
Execution Time: 352.252 ms	

В цьому випадку запит оброблявся довше.

Висновок щодо роботи індексу Hash

Hash індекси гарно працюють коли потрібно знайти точні збіги, а при пошуку часткових збігів використання індексу не пришвидшує, а інколи навіть уповільнює обробку запиту.

Мій телеграм: <https://t.me/Redivius>

Мій ГітХаб: <https://github.com/AntonPetrash/Lab2-BD.git>