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”
тема “Засоби оптимізації роботи СУБД PostgreSQL”

Виконав(ла)
студент(ка) II курсу
групи КП-01
Балагуш Ольга Іванівна
варіант №2

Перевірив
“____” “_____” 20___ р.
викладач
Радченко Костянтин
Олександрович

Київ 2021

Мета роботи

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

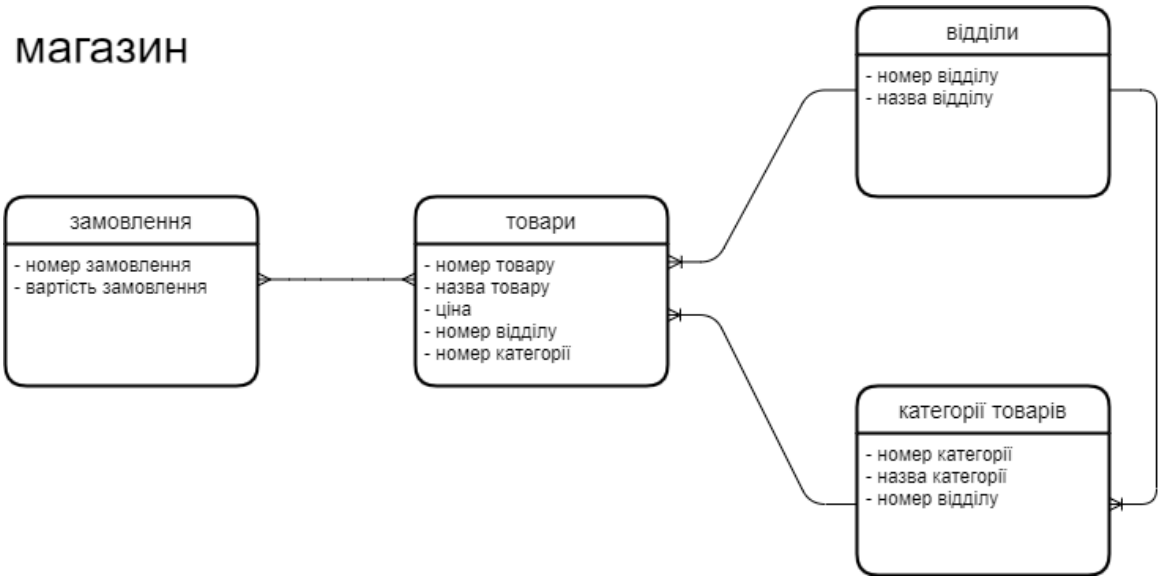
Постановка завдання

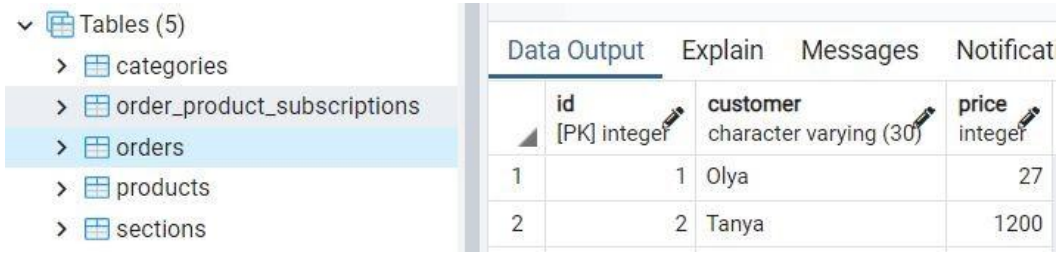
1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.

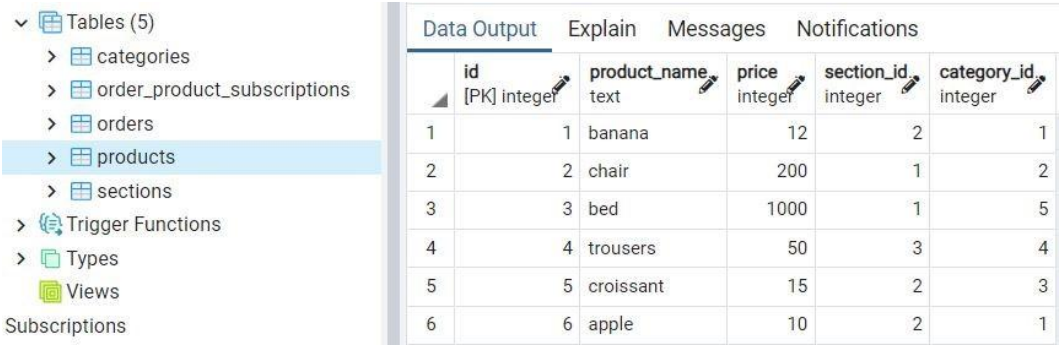
Номер варіанта №	Види індексів	Умови для тригера
2	*Hash, BRIN*	*after insert, update*

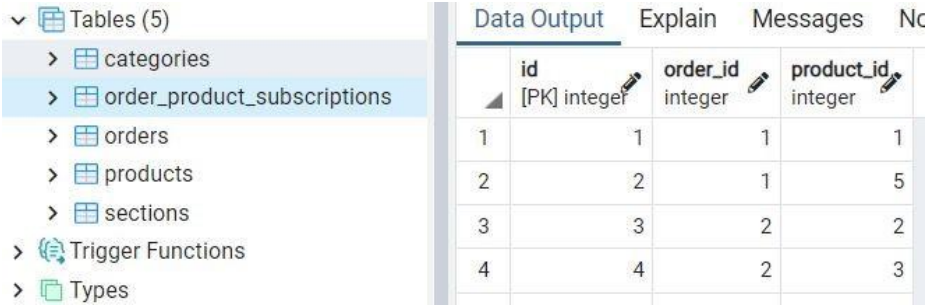
Копії екрану, що підтверджують вимоги завдання:

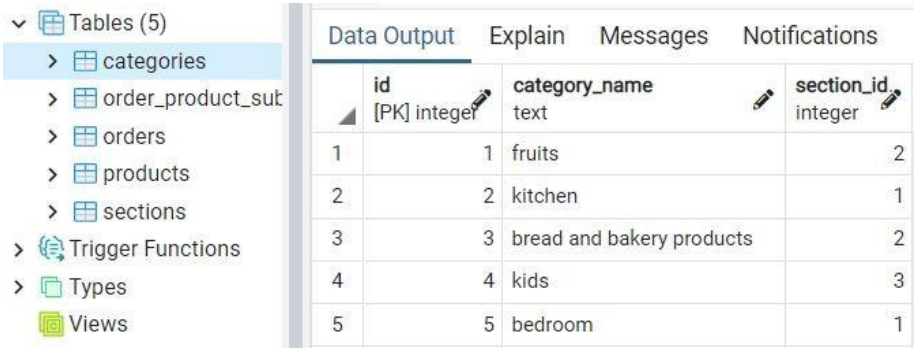
1. Схеми бази даних у вигляді таблиць і зв’язків між ними:











2. Класи ORM, що відповідають таблицям бази даних:

```python
class Section(Base):
    __tablename__ = 'sections'
    id = Column(Integer, primary_key=True, unique=True, nullable=False)
    section_name = Column(String(50))
    __table_args__ = {'extend_existing': True}

class Category(Base):
    __tablename__ = 'categories'
    id = Column(Integer, primary_key=True, unique=True, nullable=False)
    category_name = Column(String(50))
    section_id = Column(Integer, ForeignKey('sections.id'))
    __table_args__ = {'extend_existing': True}

class Subscription(Base):
    __tablename__ = 'subscriptions_o_p'
    id = Column(Integer, primary_key=True, unique=True, nullable=False)
    order_id = Column(Integer, ForeignKey('orders.id'))
    product_id = Column(Integer, ForeignKey('products.id'))
    __table_args__ = {'extend_existing': True}

class Order(Base):
    __tablename__ = 'orders'
    id = Column(Integer, primary_key=True, unique=True, nullable=False)
    customer = Column(String(50))
    price = Column(Integer)
    __table_args__ = {'extend_existing': True}

class Product(Base):
    __tablename__ = 'products'
    id = Column(Integer, primary_key=True, unique=True, nullable=False)
    product_name = Column(String(50))
    price = Column(Integer)
    category_id = Column(Integer, ForeignKey('catecories.id'))
    __table_args__ = {'extend_existing': True}
```

3. Приклади команд:

```python
def create(self, id, section_name):
    if (id < 1):
        print('Error with input!')
        return
    try:
        session = Session()
        session.add(Section(
            id = id,
            section_name = section_name
        ))
        session.commit()
        print("Entity inserted")

    except (Exception, Error) as error:
        print("Error with PostgreSQL", error)
    finally:
        print()
```

```python
def update(self, id, section_name):
    if (id < 1):
        print('Error with input!')
        return
    try:
        i = session.query(Section).get(id)
        i.section_name = section_name
        session.add(i)
        session.commit()
        print("Entity updated")
    except (Exception, Error) as error:
        print("Error with PostgreSQL", error)
    finally:
        print()

def delete(self, id):
    if (id < 1):
        print('Error with input!')
        return
    try:
        i = session.query(Section).get(44)
        session.delete(i)
        session.commit()
        print("Entity deleted")
    except (Exception, Error) as error:
        print("Error with PostgreSQL", error)
    finally:
        print()
```

4. Команди створення індексів, тексти, результати:

    a. Hash

```python
class Index:

    def test(self):
        start = timeit.timeit()
        try:
            connection = psycopg2.connect(user="postgres",
                                          password="1a2s3d4f",
                                          host="localhost",
                                          port="5433",
                                          database="lab1shop")
            cursor = connection.cursor()
            selecr_query = """CREATE INDEX ON Sections USING HASH(id);"""
            cursor.execute(selecr_query)
            connection.commit()
            print("Result", cursor.fetchall())
        except (Exception, Error) as error:
            print("Error with PostgreSQL", error)
        finally:
            if connection:
                cursor.close()
                connection.close()
                end = timeit.timeit()
                print("Time for operation " + str(end - start))
```

Хеш-індекси відрізняються за роботою від інших типів індексів, оскільки вони зберігають значення, а не покажчики на записи, розташовані на диску. Це забезпечує швидкий пошук і вставку в індекс, тому хеш-індекси часто використовуються як первинні ключі або унікальні ідентифікатори.

b. BRIN

```
class Index:

    def test(self):
        start = timeit.timeit()
        try:
            connection = psycopg2.connect(user="postgres",
                                          password="1a2s3d4f",
                                          host="localhost",
                                          port="5433",
                                          database="lab1shop")
            cursor = connection.cursor()
            selecr_query = """CREATE INDEX ix_id ON Categories USING BRING(id);"""
            cursor.execute(selecr_query)
            connection.commit()
            print("Result", cursor.fetchall())
        except (Exception, Error) as error:
            print("Error with PostgreSQL", error)
        finally:
            if connection:
                cursor.close()
                connection.close()
                end = timeit.timeit()
                print("Time for operation " + str(end - start))
```

Використання BRIN поверне всі кортежи на всіх сторінках у певному діапазоні. Таким чином, індекс має втрати, і потрібна додаткова робота для подальшої фільтрації записів. Проте є такі переваги, як: оскільки зберігається лише підсумкова інформація про діапазон сторінок, індекси BRIN зазвичай дуже малі порівняно з іншими; втратою BRIN можна контролювати, вказуючи сторінки на діапазон; розвантажує роботу з підсумовування до вакуумування або автоматичного вакуумування. Таким чином, накладні витрати на обслуговування індексу на операцію транзакцій/DML мінімальні.

5. Команди, що ініціюють виконання тригера, текст тригера:

```python
#after insert, update
class Trigger:

    def create(self):
        try:
            connection = psycopg2.connect(user="postgres",
                                          password="1a2s3d4f",
                                          host="localhost",
                                          port="5433",
                                          database="lab1shop")
            cursor = connection.cursor()
            query = """DROP TABLE IF EXISTS Subscriptions;
                       CREATE TABLE Subscriptions(order_id integer, product_id integer);
                       CREATE OR REPLACE FUNCTION Subscriptions() RETURNS trigger AS $BODY$
                       BEGIN
                           IF NEW.order_id IS NULL THEN
                               RAISE EXCEPTION 'Order cannot have null id';
                           END IF;
                           IF NEW.product_id IS NULL THEN
                               RAISE EXCEPTION 'Product cannot have null id';
                           END IF;
                           INSERT INTO subj_logs VALUES(NEW.order_id, NEW.product_id);
                           RETURN NEW;
                       END;
                   $BODY$ LANGUAGE plpgsql;
                   DROP TRIGGER IF EXISTS order_id ON Subscriptions;
                   CREATE TRIGGER order_id AFTER INSERT OR UPDATE ON Subscriptions
                       FOR EACH ROW EXECUTE PROCEDURE order_id();"""
            cursor.execute(query)
            connection.commit()
        except (Exception, Error) as error:
            print("Error with PostgreSQL", error)
        finally:
            if connection:
                cursor.close()
                connection.close()
```

Висновок

У даній лабораторній роботі, було здобуто практичні навички використання засобів оптимізації СУБД PostgreSQL.

За допомогою лабораторної роботи вивчено та застосовано використання тригерів та індексації.