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 “Проектування корпоративних інформаційних систем”
тема “ Засоби оптимізації роботи СУБД PostgreSQL”

Виконав(ла)
студент(ка) III курсу
групи КП - 81
Ганін Владислав В’ячеславович
(*прізвище, ім’я, по батькові*)

Перевірив
“____” “_____” 20___ р.
викладач
Радченко К.О.
(*прізвище, ім’я, по батькові*)

Київ 2020

ВАРІАНТ 3

1. **Метою роботи** є здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.
2. **Види індексів** по варіанту GIN, Hash.
3. **Умови для тригера** before delete, update.
4. Для виконання 3 завдання було розширено модель бази даних до 6 сутностей.

1) Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).

```python
class Warehouse(Base):
    __tablename__ = 'warehouses'

    num = Column(Integer, primary_key=True)
    address = Column(String, nullable=False)
    phone_number = Column(String(15), nullable=False)
    city_id = Column(Integer, ForeignKey('cities.id', ondelete='restrict', onupdate='restrict'), nullable=False)

    city = relationship("City", backref="warehouses")

    def __str__(self):
        return f"Warehouse [num={self.num}, address={self.address}, " \
               f"phone_number={self.phone_number}, city_id={self.city_id}]"
```

Рис.1.1 Змінена модель Warehouse

```python
class Reweightings(Base):
    __tablename__ = 'reweightings'

    id = Column(Integer, primary_key=True)
    weight_before = Column(Integer, nullable=False)
    weight_after = Column(Integer, nullable=False)
    date_inspection = Column(DateTime, nullable=False)
    parcel_id = Column(Integer, ForeignKey('goods.id', onupdate='restrict', ondelete='restrict'), nullable=False)

    parcel = relationship("Goods", backref="reweightings")

    __table_args__ = (
        Index(
            'date_inspection_index',
            date_inspection,
            postgresql_using='hash'
        ),
    )

    def __str__(self):
        return f"Reweightings [id={self.id}, weight_before={self.weight_before}, weight_after={self.weight_after}, " \
               f"date_inspection={self.date_inspection}, parcel_id={self.parcel_id}]"
```

Рис.1.2 Змінена модель Reweightings

```python
class Contragent(Base):
    __tablename__ = 'contragents'

    ipn = Column(Integer, primary_key=True, autoincrement=False)
    name = Column(String, nullable=False)
    phone_number = Column(String(15), nullable=False)

    def __str__(self):
        return f"Contragent [ipn={self.ipn}, name={self.name}, phone_number={self.phone_number}]"

    @staticmethod
    def get_distinct_names(session: Session):
        return [name for (name,) in session.query(Contragent.name).all()]
```

Рис.1.3 Змінена модель Contragent

```python
class City(Base):
    __tablename__ = 'cities'

    id = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)

    def __str__(self):
        return f"City [id={self.id}, name={self.name}]"
```

Рис.1.4 Змінена модель City

```python
class Goods(Base):
    __tablename__ = 'goods'

    id = Column(Integer, primary_key=True)
    height = Column(Integer, nullable=False)
    width = Column(Integer, nullable=False)
    depth = Column(Integer, nullable=False)
    weight = Column(Integer, nullable=False)
    description = Column(Text)
    invoice_num = Column(Integer, ForeignKey('invoices.num', onupdate='restrict', ondelete='restrict'), nullable=False)

    def __str__(self):
        return f"Goods [id={self.id}, height={self.height}, width={self.width}, depth={self.depth}, " \
               f"weight={self.weight}, description={self.description}, invoice_num={self.invoice_num}]"
```

Рис.1.5 Змінена модель Goods

```python
class Invoice(Base):
    __tablename__ = 'invoices'

    num = Column(Integer, primary_key=True)
    date_departure = Column(Date, nullable=False)
    date_arrival = Column(Date)
    shipping_cost = Column(DECIMAL, nullable=False)
    sender_ipn = \
        Column(Integer, ForeignKey('contragents.ipn', onupdate='restrict', ondelete='restrict'), nullable=False)
    recipient_ipn = \
        Column(Integer, ForeignKey('contragents.ipn', onupdate='restrict', ondelete='restrict'), nullable=False)
    warehouse_dep_num = \
        Column(Integer, ForeignKey('warehouses.num', onupdate='restrict', ondelete='restrict'), nullable=False)
    warehouse_arr_num = \
        Column(Integer, ForeignKey('warehouses.num', onupdate='restrict', ondelete='restrict'), nullable=False)

    sender = relationship("Contragent", backref="invoices_outbox", foreign_keys=[sender_ipn])
    recipient = relationship("Contragent", backref="invoices_inbox", foreign_keys=[recipient_ipn])
    warehouse_arrival = relationship("Warehouse", backref="invoices_arriving", foreign_keys=[warehouse_arr_num])
    warehouse_departure = relationship("Warehouse", backref="invoices_departing", foreign_keys=[warehouse_dep_num])

    def __str__(self):
        return f"Invoice [num={self.num}, date_departure={self.date_departure}, date_arrival={self.date_arrival}, " \
               f"shipping_cost={self.shipping_cost}, sender_ipn={self.sender_ipn}, " \
               f"recipient_ipn={self.recipient_ipn}, warehouse_dep_num={self.warehouse_dep_num}, " \
               f"warehouse_arr_num={self.warehouse_arr_num}]"
```

Рис.1.6 Змінена модель Invoice

2) Створити та проаналізувати різні типи індексів у PostgreSQL.

```
__table_args__ = (
    Index(
        'goods_descriptions_index',
        func.to_tsvector('english', description),
        postgresql_using='gin'
    ),
    Index(
        'height_index',
        height,
        postgresql_using='hash'
    ),
    Index(
        'invoice_num_index',
        invoice_num,
        postgresql_using='hash'
    ),
)
```

Рис.2.1 Створення індексів для Goods

```
__table_args__ = (
    Index(
        'sender_ipn_index',
        sender_ipn,
        postgresql_using='hash'
    ),
    Index(
        'shipping_cost_index',
        shipping_cost,
        postgresql_using='hash'
    ),
)
```

Рис.2.2 Створення індексів для Invoices

```
__table_args__ = (
    Index(
        'date_inspection_index',
        date_inspection,
        postgresql_using='hash'
    ),
)
```

Рис.2.3 Створення індексів для Reweightings

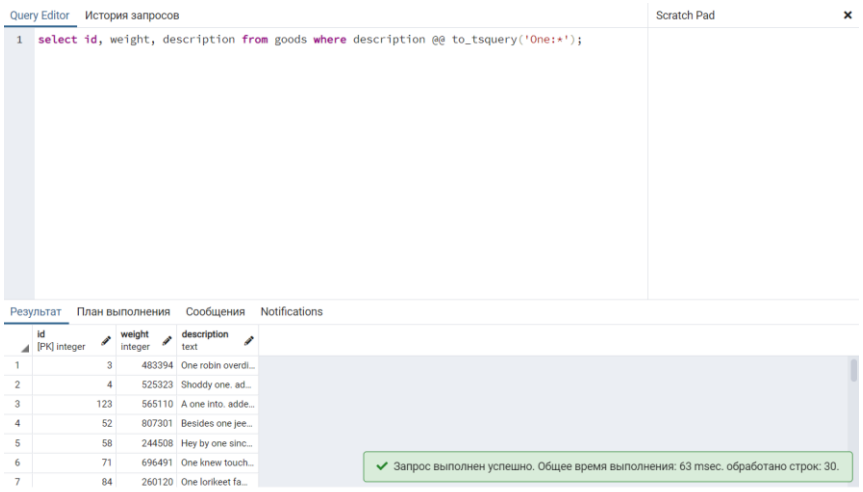

Рис.2.4 Виклик №1 з використанням індексу

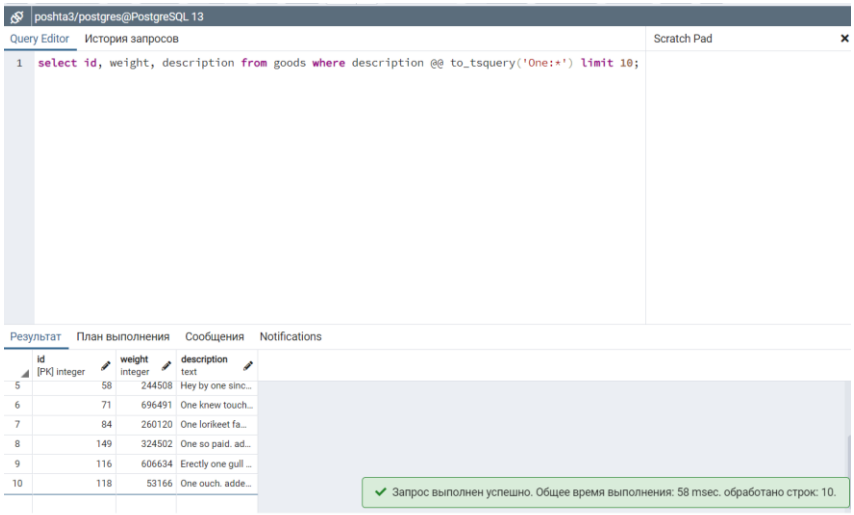

Рис.2.5 Виклик №2 з використанням індексу (фільтрування)

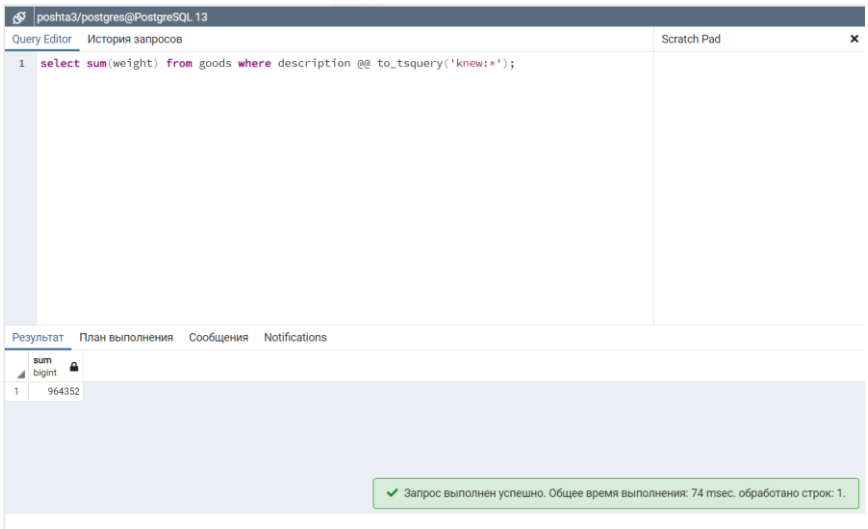

Рис.2.6 Виклик №3 з використанням індексу (агрегатна функція)

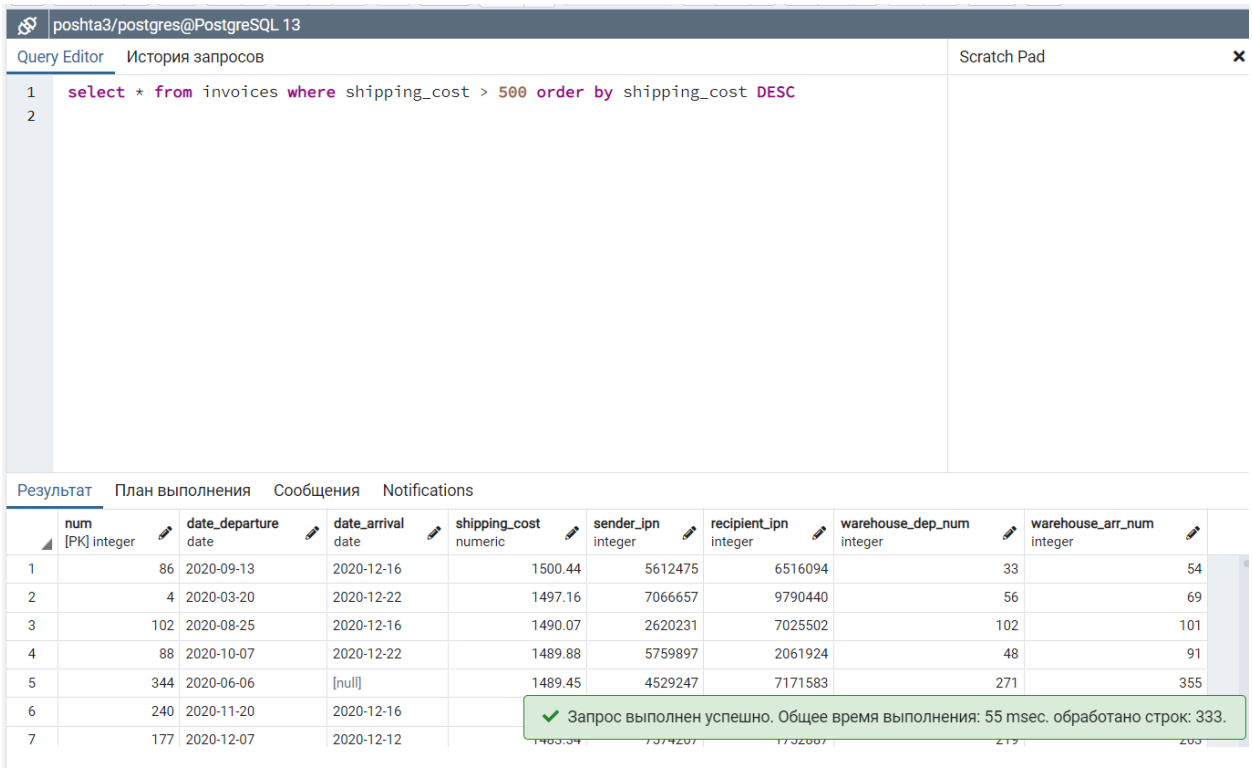

Рис.2.7 Виклик №4 з використанням індексу (сортування)

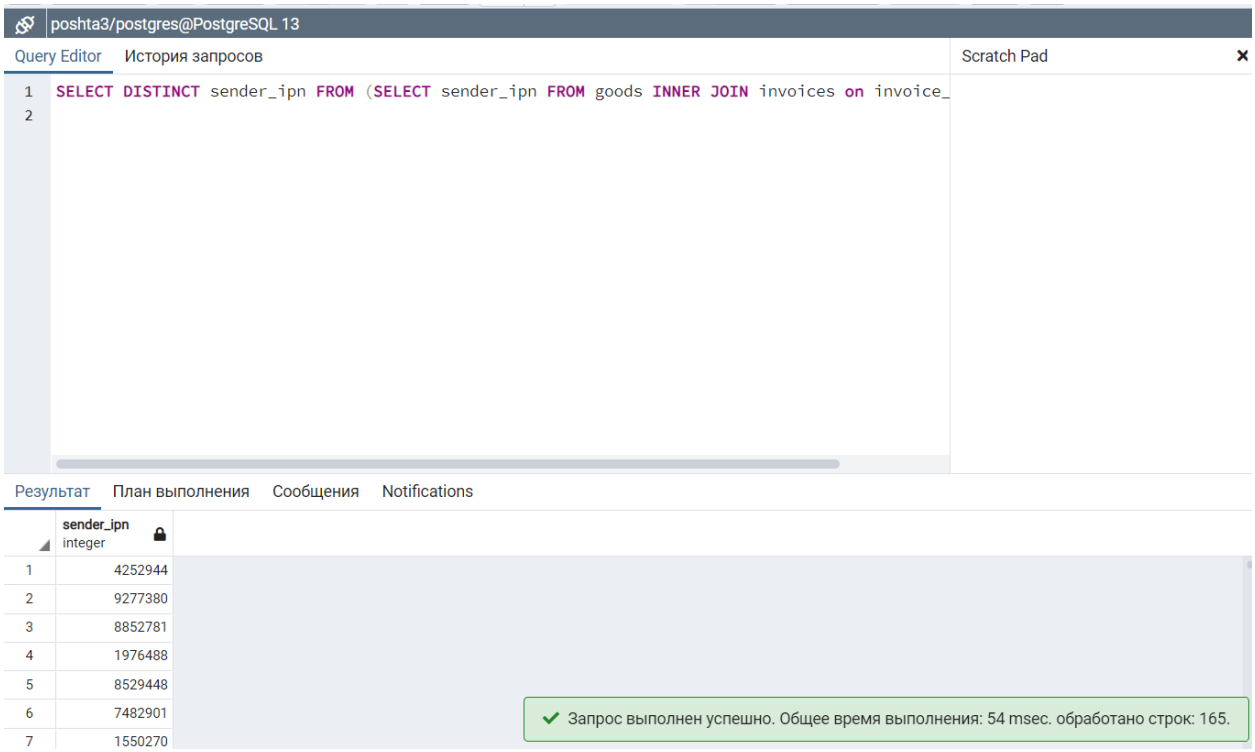

Рис.2.8 Виклик №5 з використанням індексу (групування)

3) Розробити тригер бази даних PostgreSQL.

```
-- Trigger below

CREATE OR REPLACE function goods_trigger() returns trigger as $$
    declare
        description_with_time text;
        current_word prohibited_items_dict.word%TYPE;
    begin
        if old is not null and old.weight != new.weight then
            INSERT INTO reweightings (weight_before, weight_after, date_inspection, parcel_id)
                VALUES (old.weight, new.weight, current_timestamp, new.id);
        elseif old is null then
            description_with_time = coalesce(new.description, '') || ' added to db at: ' || current_timestamp::char(19);
            UPDATE goods SET description = description_with_time WHERE id = new.id;
        end if;
        for current_word in SELECT word FROM prohibited_items_dict loop
            if old.description ilike '%' || current_word || '%' then
                raise 'This item has overcome the ban! Anxiety!';
            end if;
        end loop;
        return new;
        exception
            when no_data_found then
            when too_many_rows then
                raise 'Could not check description, try again later';
    end;
$$ language plpgsql;

DROP trigger IF EXISTS goods_helper on goods;
CREATE trigger goods_helper BEFORE DELETE OR UPDATE on goods for each row EXECUTE procedure goods_trigger();
```

Рис.3.1 Створення тригера (умовні оператори, курсорні цикли та обробка виключних ситуацій - виділені)

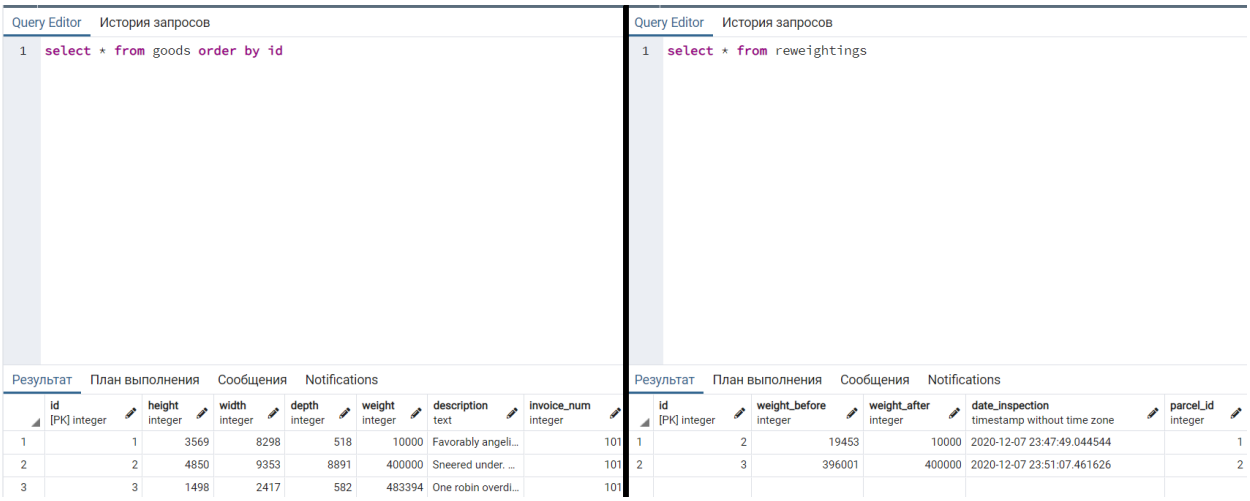

Рис.3.2 Результат оновлення goods (створення об’єкту reweighttings)

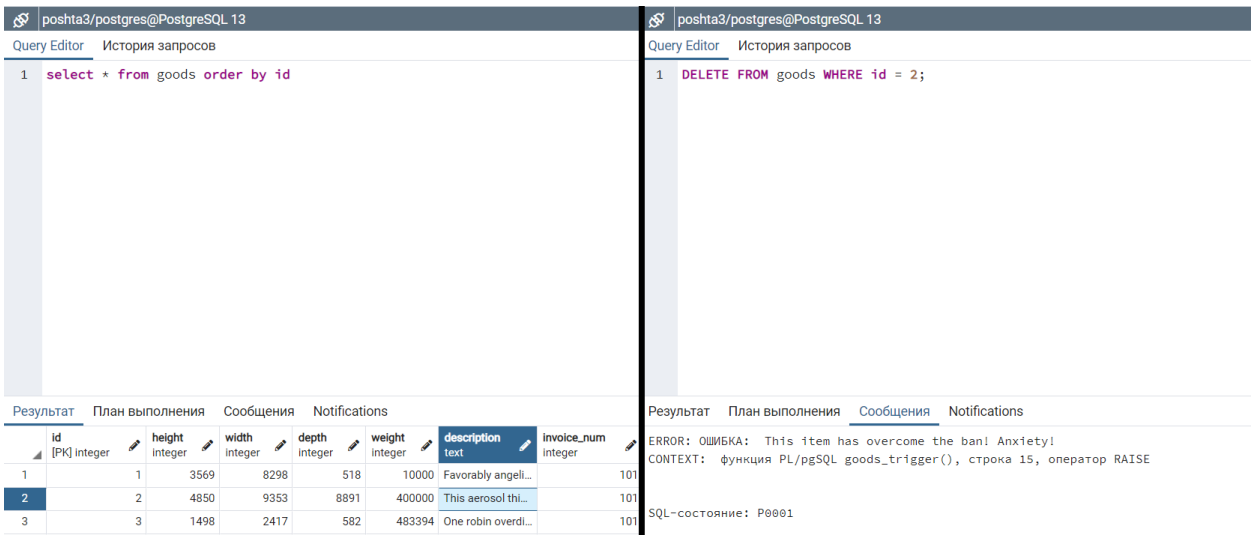

Рис.3.3 Результат видалення goods (перевірка товарів, що були перевезені)