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 1”

тема “Засоби оптимізації роботи СУБД PostgreSQL”

Виконав
студент 2 курсу
групи КП-02

Чекурда Андрій Віталійович
(*прізвище, ім’я, по батькові*)

Київ 2021

Мета роботи

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

Постановка завдання

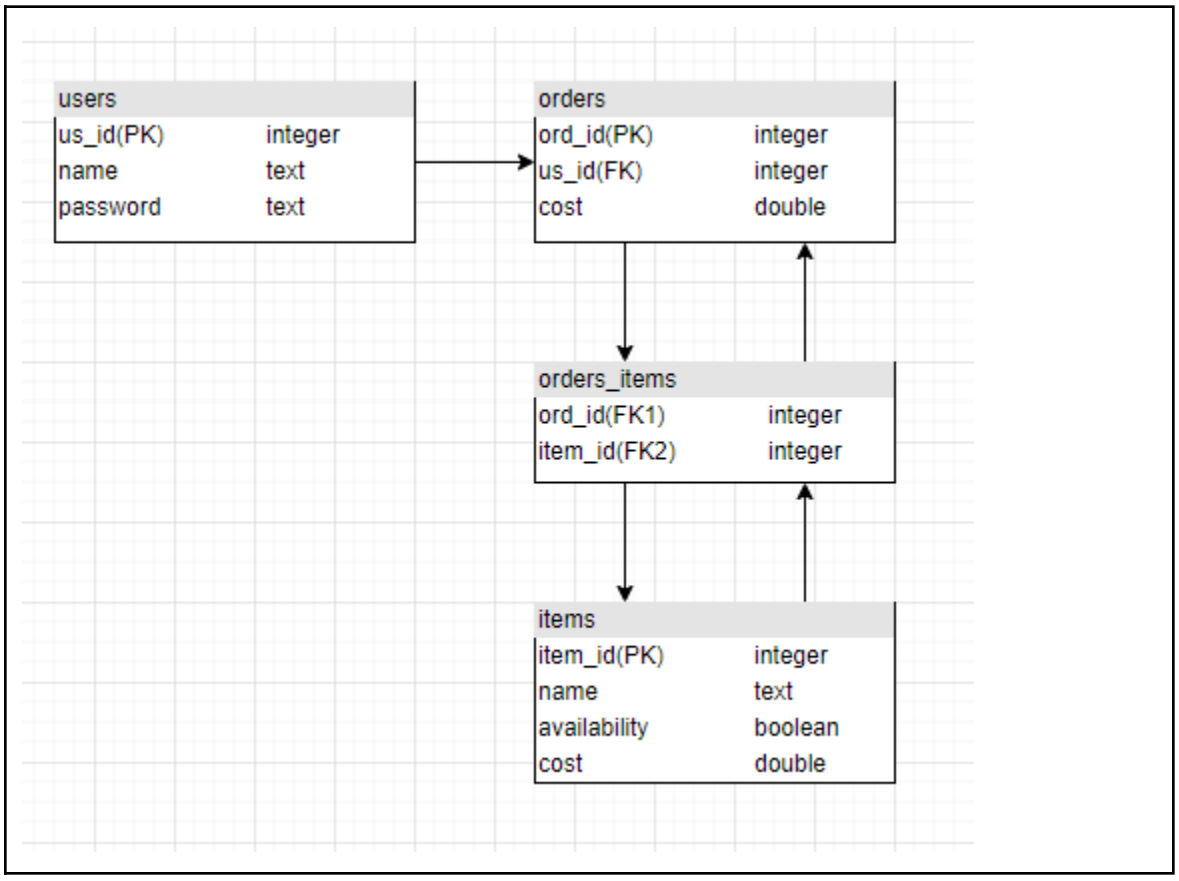
1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи No2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.

Варіант 18:

18	*BTree, GIN*	*after update, insert*

Скріншоти результатів та фрагменти коду

1. Схема бази даних



Класи ORM, що відповідають таблицям:

```
using System;
using System.Collections.Generic;

namespace lab3
{
    public partial class Item
    {
        public Item(double cost, bool availability, string name)
        {
            Cost = cost;
            Availability = availability;
            Name = name;
        }
        public Item()
        {

        }
        public int ItemId { get; set; }
        public double Cost { get; set; }
        public bool Availability { get; set; }
        public string Name { get; set; }
        public override string ToString()
        {
            return $"Id: {this.ItemId}, Name: {this.Name}";
        }
    }
}
```

```
using System;
using System.Collections.Generic;

namespace lab3
{
    public partial class Order
    {
        public int OrdId { get; set; }
        public double Cost { get; set; }
        public int UsId { get; set; }
        public virtual User Us { get; set; }
        public override string ToString()
        {
            return $"Id: {this.OrdId}, Cost: {this.Cost}";
        }
    }
}
```

```
using System;
using System.Collections.Generic;

namespace lab3
{
    public partial class User
    {
        public User()
```

```
        {
            Orders = new HashSet<Order>();
        }

        public int UsId { get; set; }
        public string Name { get; set; }
        public string Password { get; set; }

        public virtual ICollection<Order> Orders { get; set; }
        public override string ToString()
        {
            return $"Id: {this.UsId}, Name: {this.Name}";
        }
    }
}
```

Приклади запитів на основі ORM на прикладі класу ItemRepository:

```
        public object Insert(Item item)
        {
            context.Items.Add(item);
            context.SaveChanges();
            return item.ItemId;
        }
        public bool Delete(int id)
        {
            Item item = context.Items.Find(id);
            if (item == null)
            {
                return false;
            }
            else
            {
                context.Items.Remove(item);
                DeleteByItIdOrdsIts(item.ItemId);
                context.SaveChanges();
                return true;
            }
        }
        public Item GetById(int id)
        {
            Item item = context.Items.Find(id);
            if (item == null)
            {
                throw new NullReferenceException("Cannot find an object with such id.");
            }
            else
                return item;
        }
        public bool Update(Item item)
        {
            try
```

```
        {
            Item itemToUpdate = context.Items.Find(item.ItemId);
            if (itemToUpdate == null)
                return false;
            else
                itemToUpdate.Name = item.Name;
                itemToUpdate.Cost = item.Cost;
                itemToUpdate.Availability = item.Availability;
            context.SaveChanges();
            return true;
        }
        catch(Exception ex)
        {
            WriteLine(ex.Message);
            return false;
        }
    }
```

2. Команди створення індексів: (для трьох репозиторіїв відповідно)

```
    private void AddingIndexes()
    {
        this.addIndex = true;
        connection.Open();
        NpgsqlCommand command = connection.CreateCommand();
        command.CommandText = @"
            CREATE INDEX if not exists users_psw_idx ON users using GIN (password);
            CREATE INDEX if not exists users_name_idx ON users using GIN (name);
        ";
        int nChanged = command.ExecuteNonQuery();
        connection.Close();
    }
```

```
    private void AddingIndexes()
    {
        this.addIndex = true;
        connection.Open();
        NpgsqlCommand command = connection.CreateCommand();
        command.CommandText = @"
            CREATE INDEX if not exists orders_cost_idx ON orders (cost);
            CREATE INDEX if not exists orders_usId_idx ON orders (us_id);
        ";
        int nChanged = command.ExecuteNonQuery();
        connection.Close();
    }
```

```
    private void AddingIndexes()
    {
        this.addIndex = true;
        connection.Open();
        NpgsqlCommand command = connection.CreateCommand();
        command.CommandText = @"
            CREATE INDEX if not exists items_cost_idx ON items (cost);
```

```
            CREATE INDEX if not exists items_name_idx ON items using GIN (name);
        ";
        int nChanged = command.ExecuteNonQuery();
        connection.Close();
    }
```

Результати і час виконання sql запитів з індексами:

```
Enter a command: search
Enter a search subline: B
Enter measures for cost for order and item: 200,20000
Enter measures for us_id for order and item: 4,600
Enter bool value for item: true
Serch Items:
Id: 6, Name: Beta item
Id: 9, Name: Beta item
Id: 10, Name: Beta item
Id: 59, Name: BN
Id: 104, Name: SB
...
Id: 2868, Name: GXBW
Id: 2906, Name: TBGG
Id: 3119, Name: PXEB
Id: 3172, Name: YBWR
Elapsed time for item search is: 00:00:00.1797481
Serch Orders:
Id: 11, Cost: 4970
Id: 13, Cost: 1885
Id: 15, Cost: 13226
Id: 25, Cost: 376
Id: 32, Cost: 9571
Id: 43, Cost: 7330
...
Id: 591, Cost: 16906
Id: 598, Cost: 13446
Elapsed time for order search is: 00:00:00.0402294
Serch Users:
Id: 107, Name: JBWF
Id: 154, Name: HKXB
Id: 172, Name: IRPB
Id: 232, Name: PDBL
Id: 323, Name: VBWS
Id: 436, Name: REBH
Id: 480, Name: DPBB
Id: 489, Name: HBHG
...
```

```
Id: 2030, Name: YTFB
Id: 2049, Name: BMRY
Elapsed time for user search is: 00:00:00.0470458
Elapsed time for all search is: 00:00:03.3109610
Enter a command:
```

Отже, на скріншотах ми бачимо, що загалом швидкість підвищилась, оскільки на звичайний запит пошуку без індексів витрачається близько 1.4 секунди, з індексами до 2 сотих секунди, однак весь пошук одночасно у всіх таблиць сповільнює комп'ютер, тому весь пошук трохи сповільнюється. Індекси прискорюють пошук саме через створення спеціальної структури таблиці, по якій легше здійснювати пошук, наприклад збалансованого дерева.

3. Текст команд, за допомогою яких створюється тригер та його функція:

```
        private void CreateTrigger()
        {
            connection.Open();
            NpgsqlCommand command = connection.CreateCommand();
            command.CommandText = @"CREATE TRIGGER items_afup_trg AFTER UPDATE ON items FOR EACH ROW
EXECUTE PROCEDURE afup();";
            int nChanged = command.ExecuteNonQuery();
            connection.Close();
        }
        private void CreateTriggerFunction()
        {
            connection.Open();
            NpgsqlCommand command = connection.CreateCommand();
            command.CommandText =
            @"
                CREATE OR REPLACE FUNCTION afup() RETURNS TRIGGER AS $items_afup_trg$
                    BEGIN
                        IF (NEW.name IS NULL) THEN
                            RAISE EXCEPTION 'Name cannot be null';
                        ELSIF NEW.cost IS NULL THEN
                            RAISE EXCEPTION 'Cost cannot be null';
                        ELSIF NEW.cost < 0 THEN
                            RAISE EXCEPTION '% cannot have a negative cost', NEW.name;
                        END IF;
                        RETURN NEW;
                    END;
                $items_afup_trg$ language plpgsql;
            ";
            int nChanged = command.ExecuteNonQuery();
            connection.Close();
            if(!addTrigger)
            {
                CreateTrigger();
                addTrigger = true;
            }
```

Скріншоти зі змінами у таблиці даних або виведення помилки за допомогою тригера(виклик здійснюється після команди UPDATE):

Зміна даних

До:

Data Output | Explain | Messages | Notifications

	item_id [PK] integer	cost double precision	availability boolean	name character varying (100)
1	1	5000	true	Iphone 6s

Після:

	item_id [PK] integer	cost double precision	availability boolean	name character varying (100)
1	1	30000	false	New iphone 12

Виведення помилки про некоректний cost за допомогою тригера

```
Enter a command: ui1
Enter a name of updating item: dasdas
Enter a cost of updating item: -1
Enter an availability of inserting item: sadas
Availibility is wrong, enter again!
Enter an availability of inserting item: true
An error occurred while saving the entity changes. See the inner exception for details.
Was item updated successfully? - False
Enter a command:
```

Контрольні питання

1) Об’єктно-реляційна проекція - це технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних».
2) Всі індекси мають схожу будову та закладену в них логіку, однак є і певні характерні відмінності.

B-Tree - це тип, який використовується OLTP-системами в основному реалізований за замовчуванням. Індекс B-дерева - це індекс, який створюється на стовпцях, що містять дуже унікальні значення. Індекс B-Tree має впорядковані записи, коли кожен запис має значення ключа пошуку та вказівник, який посилається на заданий рядок та значення. У випадку, якщо

сервер знайде відповідне обмеження, яке стосується відповідного значення, покажчик розгортається для отримання рядка.

BRIN - техніка індексації даних, призначена для обробки великих таблиць, в яких значення індексованого стовпця має деяку природну кореляцію з фізичним положенням рядка в таблиці. Вони мають такі якості таблиць, як швидка вставка рядка, швидке створення індексу. Може використовуватися для географічних даних, тимчасових рядів, логів або історії замовлень магазину, які пишуться послідовно, тому деякі колонки (дата, час, номер) частково впорядковані вже фізично, проте водночас таблиці з такими даними зазвичай розростаються до гігантських розмірів. BRIN — не унікальний індекс, тому не може використовуватися як індекс первинного ключа.

GIN index - реалізація зворотного індексу, що використовується в СУБД PostgreSQL, зокрема, для повнотекстового пошуку та пошуку вмісту полів типу JSON. У структурі індексів GIN з кожною лексемою зіставляється відсортований (зберігається у формі B-дерева) список ідентифікаторів документів, в яких вона зустрічається.

Hash index - це тип індексу, який найчастіше використовується в управлінні даними. Зазвичай він створюється в стовпці, який містить унікальні значення, такі як первинний ключ або адреса електронної пошти. Основною перевагою використання хеш-індексів є їх швидка продуктивність. Хеш-індекси дозволяють швидко знайти дані, що зберігаються в таблицях. Вони працюють, створюючи ключ індексу зі значення, а потім знаходять його на основі отриманого хешу. Це корисно, коли є багато вхідних даних із подібними значеннями або дублікатами, оскільки потрібно лише порівнювати ключі, а не переглядати всі записи.

3) Тригери застосовуються для забезпечення цілісності даних і реалізації складної бізнес-логіки. Тригер запускається сервером автоматично при спробі зміни даних у таблиці, з якою він пов'язаний. Функції тригера служать інструментарієм виконання цієї логіки, закладеної в тригері.

Висновки

Виконавши цю лабораторну роботу, я отримав навички побудови ORM системи бази даних, також здобув засоби оптимізації СУБД за допомогою індексів. Ще однією з набутих навичок після виконання завдання став тригер

та його використання для контролю правильності введених даних перед або після виконання базових CRUD операцій. Сама лабораторна робота була виконана на мові c# з використанням Npgsql для взаємодії з Postgresql та EntityFramework для побудови ORM системи.