МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

КУРСОВА РОБОТА

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: **Система аналізу рентабельності предметів одягу в інтернет-магазині**

Студент
групи КП-02 **Кривошеєва Валерія Валеріївна** _____
(підпис)
(ПІБ)

Викладач
к.т.н, доцент кафедри
СПіСКС **Радченко К.О.** _____
(підпис)

Захищено з оцінкою _____

Київ – 2021

Анотація

Метою розробки даного курсового проєкту є набуття практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобуття навичок оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проєктної документації.

Темою даного курсового проекту є створення системи аналізу рентабельності предметів одягу в інтернет-магазині.

Курсова робота складається з бази даних онлайн-магазину (предметів одягу, клієнтів, відгуків та замовлень) та консольного додатку для взаємодії з цією базою даних, написаного мовою програмування C#, який дозволяє виконувати певні операції над нею (читання, оновлення, запис, видалення, псевдовипадкова генерація записів, статистичний аналіз даних тощо).

Результатами розробки цього проєкту став консольний додаток для роботи з наближеною до реальної базою даних інтернет-магазину, що також забезпечує її оновлення та підтримання актуальною інформацією. Крім цього, було здобуто навички розробки програмного забезпечення, що взаємодіє з реляційними базами даних.

Зміст

Вступ

В сучасному світі технологій, підприємцям, що мають інтернет-магазини, та їх аналітикам дуже важливо отримувати інформацію про споживання товарів, вподобання клієнтів та свою цільову аудиторію. Напевно кожен інтернет-магазин наразі має власну базу даних з інформацією про товари, замовлення, клієнтів тощо. Проте, крім цього, важливо аналізувати всю цю інформацію, оскільки в іншому разі це може призводити до зниження прибутків та неефективного ведення бізнесу.

Отже, задля вирішення цієї проблеми, було розроблено додаток, який має на меті вирішення проблем аналізу, керування та фільтрації даних, що містяться в базах даних інтернет-магазинів.

При розробленні додатку було створено базу даних, яка має такі сутності:

- Предмети одягу
- Клієнти
- Замовлення
- Відгуки на товари

Її було спроєктовано відповідно до третьої нормальної форми. Швидкість її роботи було оптимізовано шляхом додавання індексів. Безпеку даних при аварійних ситуаціях забезпечено механізмами резервного копіювання та реплікації. Також, за допомогою розробленого додатку можна отримувати певні статистичні дані у візуалізованому вигляді.

Крім цього, було створено підсистему попередньої обробки даних, що включає в себе засоби генерації даних. Таким чином, з’являється можливість одразу протестувати розроблений додаток на великій кількості сутностей та впевнитись в його ефективності.

1. Аналіз інструментарію для виконання курсової роботи

1. Обґрунтування вибору СУБД.

   Для виконання даної роботи в якості СУБД було обрано PostgreSQL.

   Такий вибір було зроблено на основі наступних її переваг:

   - об’єктно-реляційна СУБД;
   - підтримка БД необмеженого розміру;
   - велика кількість підтримуваних типів даних;
   - легка розширюваність;
   - надійні механізми реплікації;
   - цілісність, узгодженість та доступність даних;
   - висока продуктивність.

2. Обґрунтування вибору бібліотек.

   Для взаємодії з базою даних було обрано бібліотеку Npgsql.EntityFrameworkCore, оскільки вона має такі переваги:

   - зручна для використання у мові програмування C#;
   - розроблена спеціально для роботи з PostgreSQL;
   - має зрозумілу й вичерпну документацію;
   - найпопулярніша для взаємодії з PostgreSQL у мові програмування C#;
   - пришвидшує процес розробки та створення запитів до БД.

Для візуалізації результатів статистичного аналізу даних було обрану бібліотеку ScottPlot, вона має такі переваги:

- більшість графіків можна створити за допомогою незначної кількості коду;
- є можливість налаштувати за бажанням додаткові аргументи для графіків;

- наявна вичерпна документація з великою кількістю прикладів побудови різних графічних об’єктів;
- дозволяє легко відображати великі набори даних.

2. Структура бази даних

База даних проєкту містить 5 таблиць:

1. Garments - предмети одягу.
    - id (integer) - автоінкрементний унікальний ідентифікатор;
    - name (text) - назва продукту;
    - brand (text) - назва бренду;
    - cost (integer) - вартість продукту;
    - manufacture_country (text) - назва країни-виробника.
2. Clients - клієнти інтернет-магазину.
    - id (integer) - автоінкрементний унікальний ідентифікатор;
    - fullname (text) - повне ім’я клієнта;
    - email (text) - електронна пошта клієнта;
    - birthday_date (date) - дата народження клієнта.
3. Orders - замовлення клієнтів в інтернет-магазині.
    - id (integer) - автоінкрементний унікальний ідентифікатор;
    - shipping_method (text) - спосіб доставки замовлення;
    - created_date (date) - дата створення замовлення;
    - client_id (integer) - зовнішній ключ до таблиці Clients, який вказує на замовника, прив’язаний до поля id таблиці Clients.
4. Reviews - відгуки щодо предметів одягу.
    - id (integer) - автоінкрементний унікальний ідентифікатор;
    - opinion (text) - текст відгуку;
    - rating (integer) - оцінка від 1 до 10 предмету одягу;
    - posted_at (date) - дата написання відгуку;
    - client_id (integer) - зовнішній ключ до таблиці Clients, який вказує на автора відгуку, прив’язаний до поля id таблиці Clients.

- garment_id (integer) - зовнішній ключ до таблиці Garments, який вказує на предмет одягу, що оцінюється, прив’язаний до поля id таблиці Garments.

5. Order_items - таблиця, створена для приведення БД до третьої нормальної форми, є допоміжною таблицею для реалізації зв’язку Many to Many між таблицями Orders та Garments.
    - id (integer) - автоінкрементний унікальний ідентифікатор;
    - order_id (integer) - зовнішній ключ до таблиці Orders, який вказує на замовлення, прив’язаний до поля id таблиці Orders.
    - garment_id (integer) - зовнішній ключ до таблиці Garments, який вказує на предмет одягу, що міститься в замовленні, прив’язаний до поля id таблиці Garments.

3. Опис програмного забезпечення

3.1 Загальна структура програмного забезпечення

Розроблене програмне забезпечення складається з наступних компонентів:

1. База даних, що містить інформацію інтернет-магазину.
2. Засоби генерації даних до таблиць бази даних.
3. Засоби CRUD-функціоналу.
4. Засоби фільтрації та валідації даних.
5. Засоби пошуку даних за певними атрибутами.
6. Засоби статистичного аналізу даних.
7. Засоби візуального зображення даних.
8. Засоби реплікації.
9. Засоби резервування та відновлення даних.

3.2 Опис модулів програмного забезпечення

Розроблене програмне забезпечення складається з таких модулів:

1. Model

   Цей модуль напряму взаємодіє з базою даних. В цьому модулі знаходяться ORM класи сутностей, клас контексту, що відповідає за ORM взаємодію з базою та її схему та клас сервісу, що об’єднує попередньо перераховані класи.

2. View

   Даний модуль відповідає за консольний інтерфейс взаємодії з користувачем, а саме вивід певної інформації в консоль в залежності від запиту та результату його обробки.

3. Controller

   Цей модуль пов’язує model та view. Він отримує інформацію з Model за запитом користувача та передає її для відображення у View.

4. DataGenerator

   Це модуль генерації сутностей у таблиці бази даних. Генерація відбувається псевдовипадково для деяких сутностей, для інших - з датасетів. Є підсистемою розробленого програмного забезпечення.

5. Visualization

   Це модуль, що використовується для візуалізації отриманих статистичних даних. За допомогою нього будуються графіки, діаграми та створюється звіт про певний предмет одягу.

3.3 Опис основних алгоритмів роботи

Процес генерації даних був влаштований так, щоб згенеровані дані були якомога більш адекватними для певної предметної галузі. Генерація здійснювалась на основі датасетів, а також псевдовипадковою генерацією сутностей.

Для полегшення взаємодії з базою даних та структуруванням запитів до неї було розроблено спеціальні репозиторії, які були об’єднані у клас сервісу.

Дані статистичного аналізу зображуються на спеціальних графіках та діаграмах, які потім зберігаються у відповідну директорію.

Генерація звіту про певний товар відбувається програмним заповненням зразка документу, після чого він також зберігається у певну директорію.

4. Аналіз функціонування засобів реплікації

Для реплікації був використаний механізм логічної реплікації, що наданий виключно СУБД PostgreSQL. Він полягає у схемі публікація/підписник. Основна база даних надає публікацію для всіх таблиць разом з реплікаційним слотом. Додаткова, або резервна, база даних створює підписку до публікації вказаної вище та прив’язується до створеного реплікаційного слоту. Переваги такого виду реплікації:

- передача підписниками інкрементальних змін в одній базі даних чи підмножині баз даних, коли вони відбуваються;
- спрацьовування тригерів для окремих змін, коли їх отримує підписник;
- реплікація між різними основними версіями PostgreSQL;
- реплікація між екземплярами PostgreSQL на платформі Linux;
- надання доступу до реплікаційних даних іншим групам користувачів.

5. Аналіз функціонування засобів резервування/відновлення

Резервне копіювання необхідне для забезпечення безпечного та швидкого відновлення даних у разі їх пошкодження або видалення. Тип резервного копіювання, який було реалізовано - повне. Це такий вид резервного копіювання, який забезпечує створення повної копії об'єкту резервного копіювання. Цей вид дозволяє забезпечити максимальну відповідність оригіналу даних його копії. Перевага такого різновиду резервного копіювання полягає у тому, що не потрібно об’єднувати різні файли для відновлення, натомість відновлюється все з одного файлу, за рахунок чого відновлення є помітно швидшим порівняно з іншими видами резервного копіювання. Було використано вбудоване резервне копіювання системи керування базами даних PostgreSQL.

Розроблений додаток надає можливість користувачу керувати резервним копіюванням та відновленням. Файли резервних копій зберігаються у папці backup, яка знаходиться у data, у форматі backup{date}.sql, де date - дата резервної копії. Користувач може відновлювати базу даних ввівши шлях до файлу резервної копії. Після виконання відновлення, додаток необхідно перезапустити для відновлення зв’язку з сервером бази даних.

6. Аналіз результатів підвищення швидкодії запитів

З метою підвищення швидкодії запитів для отримання деяких даних було використано індексування деяких полів трьох таблиць. Типи індексування, які було використано - HASH та BTREE. Індексування HASH було застосовано до таких полів: поле fullname таблиці Clients, поля brand таблиці Garments, а BTREE до полів rating, posted_at таблиці Review.

У разі малої бази даних індекси є не дуже ефективними, їхні алгоритми складніші і довші ніж просто лінійний пошук, коли даних мало. У зв’язку з цим індекси і застосовуються лише для великих баз даних. Запити створення індексів:

1. CREATE INDEX cl_fullname ON clients USING hash(fullname)
2. CREATE INDEX gm_brand ON garments USING hash(brand)
3. CREATE INDEX IF NOT EXISTS rw_index ON reviews USING btree(rating, posted_at)

Результати ефективності використання наведених індексів буде наведено в додатках. Проте, одразу можна затвердити, що індекси справді підвищують швидкодію запитів за великої кількості сутностей.

7. Опис результатів аналізу предметної галузі

У розробленому консольному додатку наявний наступний аналіз даних, що містяться у базі:

- для аналізу загального рейтингу продукту серед оцінок клієнтів було використано запит, який допомагає сформувати графік розподілу оцінки продукту клієнтами. Такий графік дозволяє виявити нерентабельні товари а також звернути увагу на слабкі місця позицій з асортименту;
- для аналізу вікової категорії товару було використано запит, який у вигляді секторної діаграми демонструє зацікавленість клієнтів кожної з 5-ти вікових груп, які були обрані майже у аналогії з реальним світом. Така діаграма дозволяє визначитись з цільовою аудиторією товару та продумати маркетингові ходи на майбутнє;
- для аналізу доходів онлайн-магазину було використано запит, який збирає інформацію про всі продажі товарів за певний рік та формує ламану доходів. За допомогою такого графіку можна приблизно спрогнозувати подальший розвиток ситуації - будуть доходи магазину зростати чи спадати.

Додатково було реалізовано формування звіту про певну позицію з асортиментів товару онлайн-магазину. Звіт містить у собі повну інформацію про пару взуття, її середній рейтинг серед клієнтів, відгуки з найбільшою та найменшою оцінкою а також графік розподілу оцінки продукту клієнтами. Звіт заповнюється програмно шляхом архівування та розархівуванням документу-зразка та заміною відповідних значень у ньому.

Приклади графіків, діаграм та звіту наведені у додатках.

Висновки

Під час виконання даної курсової роботи я набула практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобула навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проєктної документації.

В процесі виконання даного курсового проєкту було отримано практичні навички роботи з реплікацією, створенням резервних копій та забезпеченням отримання цієї інформації.

У результаті було виконано такі задачі:

- розробка бази даних, яка відповідає третій нормальній формі;
- реплікацію було реалізовано виду logical replication;
- резервне копіювання було реалізовано повне, що дає можливість швидкого відновлення;
- було здійснено перетворення складних запитів на більш ефективні та компактні за допомогою ORM;
- була розроблена псевдовипадкова генерація для всіх таблиць, яка генерує якомога більш реалістичні значення;
- була підвищена швидкодія запитів до бази даних шляхом індексування деяких полів деяких таблиць;
- були розроблені засоби для аналізу даних, які надають можливість виводити графічне представлення його результату для наочності висновків;
- був розроблений зручний консольний інтерфейс, який обробляє всі помилки, валідує дані та надає можливість виконання CRUD-операцій та фільтрації командами, а не SQL-запитами.

Література

1. PostgreSQL 10.19 Documentation [Електронний ресурс]

   Режим доступу: https://www.postgresql.org/docs/10/index.html.

2. Npgsql documentation [Електронний ресурс]

   Режим доступу: https://www.npgsql.org/.

3. Entity Framework Documentation [Електронний ресурс]

   Режим доступу: https://docs.microsoft.com/en-us/ef/

4. ScottPlot.NET documentation [Електронний ресурс]

   Режим доступу: https://scottplot.net/.

5. pgAdmin 4 4.28 documentation [Електронний ресурс]

   Режим доступу:
   https://www.pgadmin.org/docs/pgadmin4/4.28/index.html.

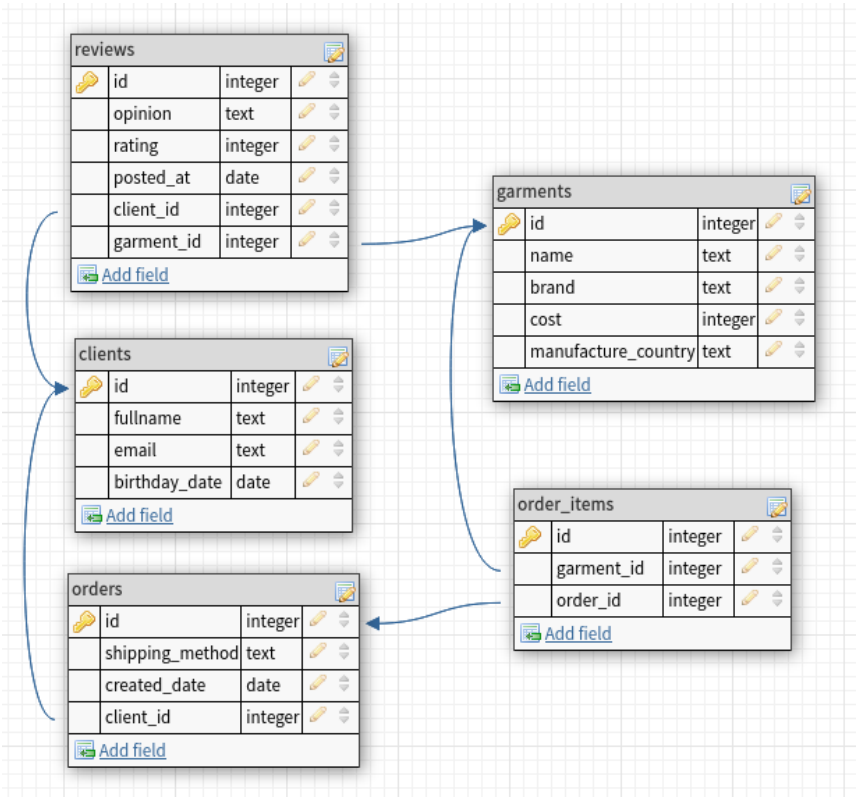
6. Logical replication

   Режим доступу:
   https://www.postgresql.org/docs/10/logical-replication.html.

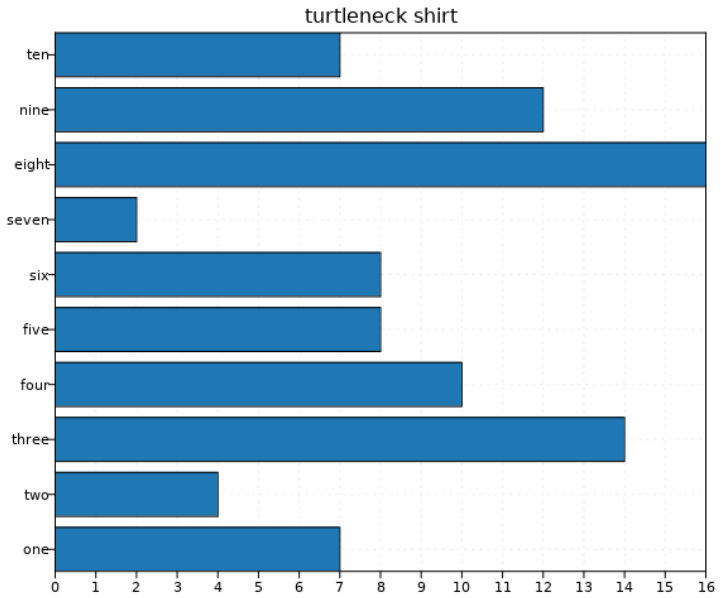
Додатки

А. Графічні матеріали

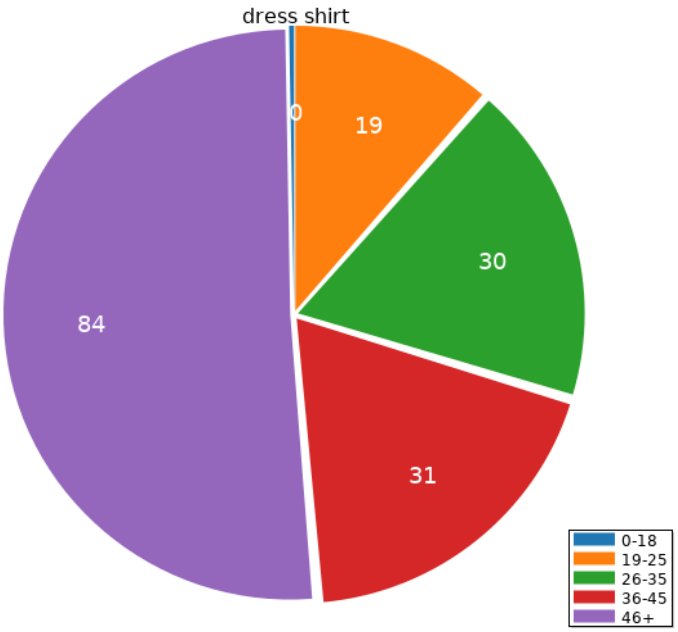
Структура бази даних:



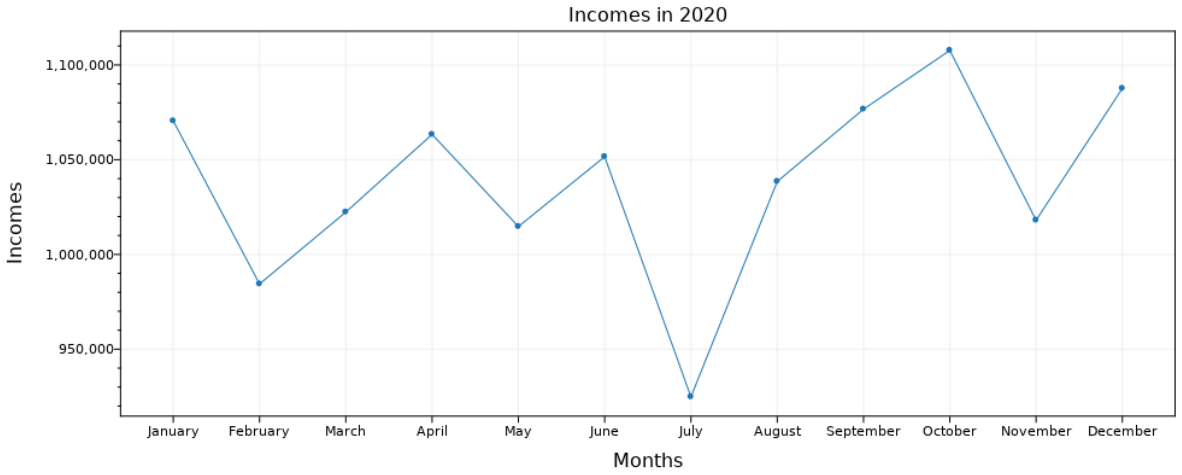
Результат аналізу загального рейтингу продукту серед оцінок клієнтів:



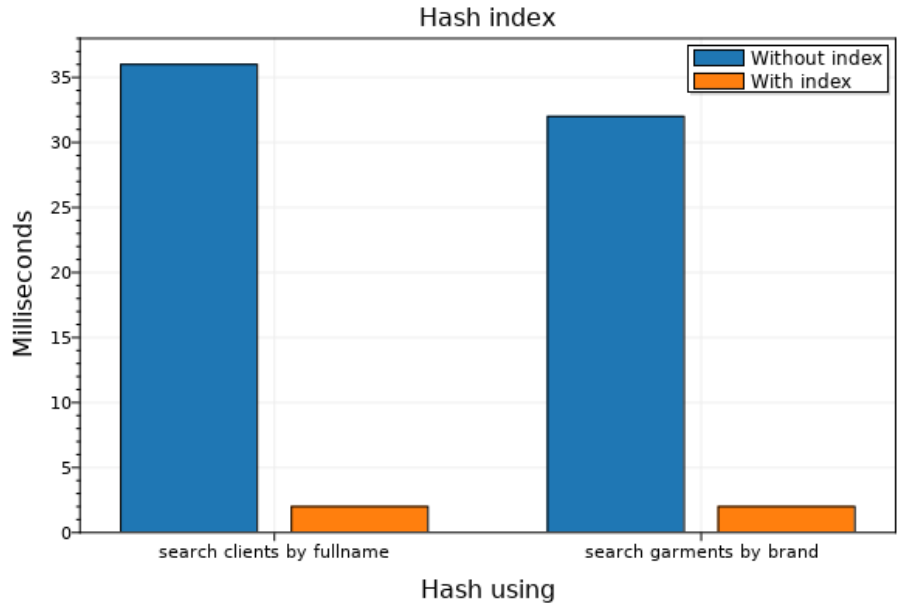
Результат аналізу вікової категорії товару:

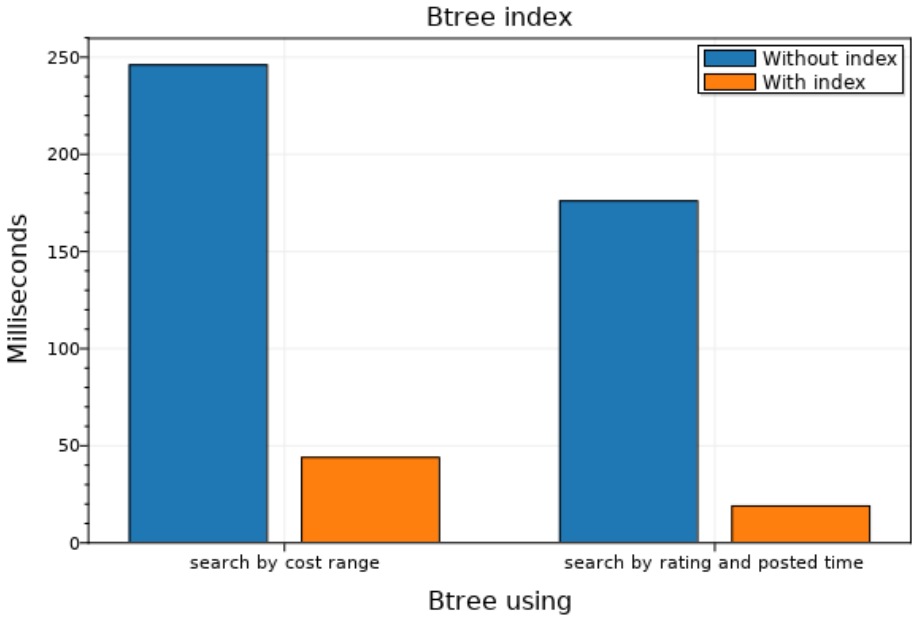


Результат аналізу доходів інтернет-магазину:

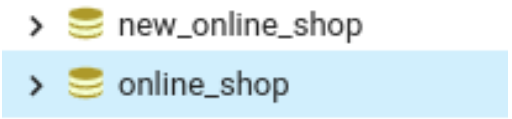


Порівняння часу виконання вказаних у відповідному розділі запитів з індексуванням та без:





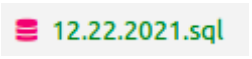
Забезпечення засобів реплікації:



Резервування даних:

```
Backup was done successfully in [2222] ms!
```

```
You can find it by directory /home/valeria/Desktop/db_course_work/data/backup/backup.sh 12.22.2021 10:20:23 AM
```



Відновлення даних:

```
Enter file path to restore from: ./../data/backup/12.22.2021.sql
UPDATE 1
 pg_terminate_backend
----------------------
 t
 t
 t
 t
 t
(5 rows)

DROP DATABASE
CREATE DATABASE
Restore was done successfully in [2363] ms from file by directory ./../data/backup/12.22.2021.sql
Please, restart program.
```

Б. Фрагменти програмного коду

Псевдовипадкова генерація сутностей

```
public void Generate(int amount)
        {
            string query = @$"CREATE TRIGGER order_items
                              AFTER INSERT ON orders
                              FOR EACH ROW EXECUTE PROCEDURE trigger();";
            context.Database.ExecuteSqlRaw(query);

            int[] clientIds = context.clients.Select(o => o.id).ToArray();
            Random rand = new Random();
            for(int i = 0; i < amount; i++)
            {
                query = @$"INSERT INTO orders (created_date,
shipping_method, client_id)
                           SELECT
                           timestamp '2018-01-01' + random() * (timestamp
'2021-12-12' - timestamp '2018-01-01'),
                           random_choice(array['by air', 'by land', 'by
sea']) as random_char,
                           {clientIds[rand.Next(0,clientIds.Length)]}";
                context.Database.ExecuteSqlRaw(query);
            }
            query = "DROP TRIGGER order_items ON orders";
            context.Database.ExecuteSqlRaw(query);
        }
```

Приклад CRUD

```
public int Insert(Garment garment)
        {
            context.garments.Add(garment);
```

```
            context.SaveChanges();
            return garment.id;
        }

        public void Update(int id, Garment garment)
        {
            var local = context.garments.Find(id);
            if (local != null)
            {
                context.Entry(local).State = EntityState.Detached;
            }
            context.Entry(garment).State = EntityState.Modified;
            context.SaveChanges();
        }
        public void DeleteById(int id)
        {
            context.garments.Remove(context.garments.Find(id));
            context.SaveChanges();
        }
        public Garment GetById(int id)
        {
            Garment result = context.garments.Find(id);
            return result;
        }
```

Резервне копіювання та відновлення

```
public void Backup()
        {
            Process process = new Process
            {
                StartInfo = new ProcessStartInfo
                {
                    FileName = "/bin/bash",
                    Arguments = $"-c
\"/home/valeria/Desktop/db_course_work/data/backup/backup.sh
{DateTime.Now.ToString().Replace('/','.')}\"",
                    UseShellExecute = false,
                    CreateNoWindow = true
                }
            };
            try
            {
                Stopwatch sw = new Stopwatch();
                sw.Start();
                process.Start();
                process.WaitForExit();
                sw.Stop();

View.OutputBackupInfo($"/home/valeria/Desktop/db_course_work/data/backup/ba
ckup.sh {DateTime.Now.ToString().Replace('/','.')}",
sw.ElapsedMilliseconds);
            }
            catch(Exception ex)
            {
                View.OutputError(ex.Message);
            }
```

```
        }
        public void Restore(string filePath)
        {
            if(!File.Exists(filePath))
            {
                View.OutputError("Entered file path doesn`t exist.");
                return;
            }
            if(!filePath.EndsWith(".sql"))
            {
                View.OutputError("Entered file path is incorrect.");
                return;
            }
            Process process = new Process
            {
                StartInfo = new ProcessStartInfo
                {
                    FileName = "/bin/bash",
                    Arguments = $"-c
\"/home/valeria/Desktop/db_course_work/data/backup/restore.sh
{filePath}\"",
                    UseShellExecute = false,
                    CreateNoWindow = true,
                    Verb = "sudo"
                }
            };
            try
            {
                Stopwatch sw = new Stopwatch();
                sw.Start();
                process.Start();
                process.WaitForExit();
                sw.Stop();
                View.OutputRestoreInfo(filePath, sw.ElapsedMilliseconds);
            }
            catch(Exception ex)
            {
                View.OutputError(ex.Message);
            }
            Environment.Exit(0);
        }
```

Аналіз річних доходів

```
public List<double> GetIncomesByYear(int year)
        {
            List<double> result = new List<double>();
            var connection = context.Database.GetDbConnection();
                try
                {
                    connection.Open();
                    using (var command = connection.CreateCommand())
                    {
                        for(int i = 1; i<12; i++)
                        {
                            command.CommandText = @$"SELECT
SUM(garments.cost) FROM garments,orders,order_items WHERE
```

```
orders.created_date >= timestamp '{year}-{i}-1' AND orders.created_date <
timestamp '{year}-{i+1}-1'
                            AND orders.id = order_items.order_id AND
order_items.garment_id = garments.id";

result.Add(Convert.ToDouble(command.ExecuteScalar()));
                        }
                        command.CommandText = @$"SELECT SUM(garments.cost)
FROM garments,orders,order_items WHERE orders.created_date >= timestamp
'{year}-12-1' AND orders.created_date < timestamp '{year+1}-1-1'
                        AND orders.id = order_items.order_id AND
order_items.garment_id = garments.id";

result.Add(Convert.ToDouble(command.ExecuteScalar()));
                    }
                }
                catch (System.Exception e) { }
                finally { connection.Close(); }
                return result;
        }
```

Створення графіка для аналізу річних доходів

```
public static void CreateIncomesChart(List<double> incomes, int year)
        {
            var plt = new ScottPlot.Plot(1000, 400);

            double[] values = incomes.ToArray();
            double[] xs = DataGen.Consecutive(12);
            string[] labels = new string[]{"January", "February", "March",
"April", "May", "June", "July", "August", "September", "October",
"November", "December"};
            plt.AddScatter(xs, values);
            plt.XAxis.ManualTickPositions(null, labels);
            plt.XTicks(xs, labels);
            plt.Title($"Incomes in {year}");
            plt.XLabel("Months");
            plt.YLabel("Incomes");

            plt.SaveFig($"./../data/Incomes in {year}.png");
        }
```

Приклад фільтрації

```
public List<Garment> GetByBrand(string brand)
        {
            List<Garment> garments = context.garments.Where(x => x.brand ==
brand).ToList();
            return garments;
        }
```