МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-05

Дипломний проект

здобувача освіти денної форми навчання
РП.05.13.000.ДП

КОВАЛЕНКО
ЄВГЕН
ОЛЕКСАНДРОВИЧ

м. Одеса
2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**
Освітня програма: «**Розробка програмного забезпечення»**
Група: **4РП-05**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Побудова і оптимізація процедур з бази даних і розміщення їх на хмарі за рахунок інструментів Azure і команд в Kudu

Проектний матеріал складається з пояснювальної записки на 46 сторінках та графічного (презентаційного) матеріалу на 10 аркушах (слайдах).

Дипломник _____ (Коваленко Є.О.)
Керівник _____ (Сологуб К.В.)

Консультанти:
з економічної частини _____ (Копайгородська Т.Г.)
з охорони праці_____ (Чорновол Н.І.)
з дотримання вимог ЄСКД _____ (Петрашова В.І.)
старший консультант _____ (Скорнякова О.В.)

До захисту допущений
Голова циклової комісії _____ (Скорнякова О.В.)
Завідувач відділення _____ (Суліма Ю.Ю.)

Захист «____»_____ 2022 р. Протокол ДКК №_____
Оцінка ДКК _____

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНАХТ»

Відділення **комп'ютерних систем** Комісія **КТ та ПІ**

Спеціальність **121 «Інженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР_____

_____

"_____"_____2022 р.

З А В Д А Н Н Я

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти Коваленко Євген Олександрович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Побудова і оптимізація процедур з бази даних і розміщення їх на хмарі за рахунок інструментів Azure і команд в Kudu

затверджена наказом по коледжу від "30" грудня 2021 р. № 306-А2-ОД

2. Термін здачі закінченого проекту (роботи) _____

3. Вихідні данні до проекту (роботи) Microsoft Visual Studio, .Net, C#, MS SQL Server, JavaScript LINQ, API, HTTP, Microsoft Edge, Postman, MySQL, СУБД

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Оптимізація збережених процедур та їх складання. Зберігання і обробка даних за рахунок інструментів Microsoft Azure. 2. Економічний розрахунок. 3. Охорона праці.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація (10 слайдів)

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний	Сологуб К.В.		
Економічний	Копайгородська Т.Г.		
Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		
Старший консультант	Скорнякова О.В.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	Розділ 1. Оптимізація збережених процедур та їх		
	складання. Зберігання і обробка даних за рахунок		
	інструментів Microsoft Azure		
2	Розділ 2. Економічний розрахунок		
3	Розділ 3. Охорона праці		
4	Розробка презентації до дипломної роботи		
5	Чистове оформлення пояснювальної записки		
6	Підготовка доповіді до захисту		
7	Отримання рецензії, відповіді на зауваження		
	Рецензента		
8	Захист роботи		

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 05.13.000 ДП ПЗ	*Дипломний проект*		
А4			*РП 05.13.000 ДП ПЗ*	*Пояснювальна*	*1*	
				записка		
				Презентація		

Ізм.	Арк.	№ докум.	Підпис	Дата	*РП 05.13.000*		
Розробив		Коваленко Є.О			Побудова і оптимізація процедур з бази даних і розміщення їх на хмарі за рахунок інструментів Azure і команд в Kudu	Літ.	Аркуш 3 / Аркушів
Перевірив		Сологуб К.В.				*Н Д П*	
Н. Контр.		Петрашова В.І				ВСП «ОТФК ОНТУ» гр.РП-05	
Затверд.		Скорнякова О.В.					

ЗМІСТ

ВСТУП

З появою магнітних дисків розпочалася історія систем керування даними у зовнішній пам'яті. До цього кожна прикладна програма, якій потрібно зберігати дані у зовнішній пам'яті, сама визначала розташування кожної порції даних на магнітній стрічці або барабані і виконувала обміни між оперативною та зовнішньою пам'яттю за допомогою програмно-апаратних засобів низького рівня (машинних команд або викликів відповідних програм операційної системи)). Такий режим роботи не дозволяє або дуже ускладнює підтримку на одному зовнішньому носії кількох архівів інформації, що довго зберігається. Крім того, кожній прикладній програмі доводилося вирішувати проблеми іменування елементів даних та структуризації даних у зовнішній пам'яті.

База даних (БД) – це структурований набір даних, що постійно зберігаються. Постійність означає, що дані не знищуються після завершення програми або сеансу користувача, в якому вони були створені.

База даних - це набір, сукупність файлів, у яких перебуває інформація. Програмна система (додаток), що забезпечує роботу з базою даних (файлами даних) називається системою управління базою даних (СУБД).

Залежно від розташування програми, яка використовує дані, самих даних, а також від способу поділу даних між декількома користувачами розрізняють локальні та віддалені бази даних.

Процедури, що зберігаються, дозволяють підвищити продуктивність, розширюють можливості програмування і підтримують функції безпеки, недоступні при використанні команд Transact-SQL, що відсилаються для обробки на сервер. Підвищується продуктивність - за рахунок локального (стосовно бази даних) зберігання, перекомпіляції вихідного тексту та кешування. Можливості програмування розширюються завдяки застосуванню таких поширених засобів програмування, як використання вхідних та вихідних параметрів, а також завдяки багаторазовому використанню

процедур. Функції безпеки мають на увазі шифрування тексту процедури та обмеження привілеїв. В результаті користувачі отримують обмежений доступ до внутрішньої структури бази даних, однак їм дозволено запускати процедури, що зберігаються, що виконують різні дії над базою даних.

МЕТА ТА ЗАВДАННЯ

Метою дипломної роботи є оптимізація обробки LINQ – запитів на рівні бази даних за рахунок розробки та оновлення існуючих запитів на рівні збережених процедур, для подальшого розміщення хмарного сервісу для швидкої їх обробки та оптимізації «перфомансу» веб орієнтованої системи.

Для досягнення цієї мети необхідно вирішити та розробити ряд дій:

1. Скласти шаблон ASP.NET для подальшої обробки запитів;

2. Зв'язати з клієнтською частиною частину запитів, які будуть зв'язуватися з клієнтською частиною для запиту;

3. Дати якісну оцінку «перфомансу» без використання процедур, що зберігаються;

4. Скласти та зв'язати збережені процедури з існуючими CRUD – операціями та оптимізувати запит;

5. Використовуючи платформу Microsoft Azure, розмістити збережені процедури та оптимізувати існуючі запити;

6. За допомогою консолі Kudu розгорнути деплоймент проекту та протестувати результат оновлення перфомансу;

1 ОПТИМІЗАЦІЯ ЗБЕРЕЖЕНИХ ПРОЦЕДУР ТА ЇХ СКЛАДАННЯ. ЗБЕРІГАННЯ І ОБРОБКА ДАНИХ ЗА РАХУНОК ІНСТРУМЕНТІВ MICROSOFT AZURE

1.1 Фізична та логічна архітектура бази даних

Фізична база даних є набір файлів, розташованих на диску. З цими файлами можна виконувати будь-які операції, дозволені для звичайних файлів: копіювання, перейменування, видалення тощо. Виконання перелічених операцій у разі потреби можливе. Фізична структура бази даних визначає кількість файлів даних та журналу транзакцій, з яких складається база даних, їх початковий та поточний розмір, положення на диску, ім'я, розширення, крок збільшення та деякі інші параметри. Ці параметри необхідні лише для правильного сприйняття бази даних SQL Server. Для користувачів, що працюють з базою даних, у переважній більшості випадків її фізична структура не має значення.

Логічна структура бази даних, що описує її об'єкти, їх поведінка і взаємодія друг з одним. Логічна структура бази даних включає системні та користувальницькі таблиці, уявлення, збережені процедури, користувачів і ролі, замовчування, обмеження цілісності та інші об'єкти.

Фізична база даних SQL Server зберігається у самостійному, унікальному кожної БД наборі файлів. Журнал транзакцій та самі дані обов'язково зберігаються окремо. Це підвищує стійкість до відмови бази даних у разі збоїв системи.

Для зберігання бази даних призначено набір файлів, персональний для будь-якої бази даних. Кожен файл може належати лише до однієї бази даних. У SQL Server існує два типи файлів бази даних:

Файли даних (data file) призначені зберігання інформації, що у таблицях бази даних. Крім того, у цих файлах також розміщені процедури, обмеження, тригери, індекси та інша інформація;

У файлах журналу транзакцій (transaction log file) SQL Server записує інформацію про хід виконання транзакцій. У них розмішається інформація про стан даних перед початком транзакції, про зміни, блоковані ресурси та інша супутня інформація.

Будь-яка база даних має містити щонайменше один файл даних та один файл журналу транзакцій, тобто. мінімальна кількість файлів, що становлять базу даних, дорівнює 2. При необхідності адміністратор може додавати нові файли даних або файли журналу транзакцій.

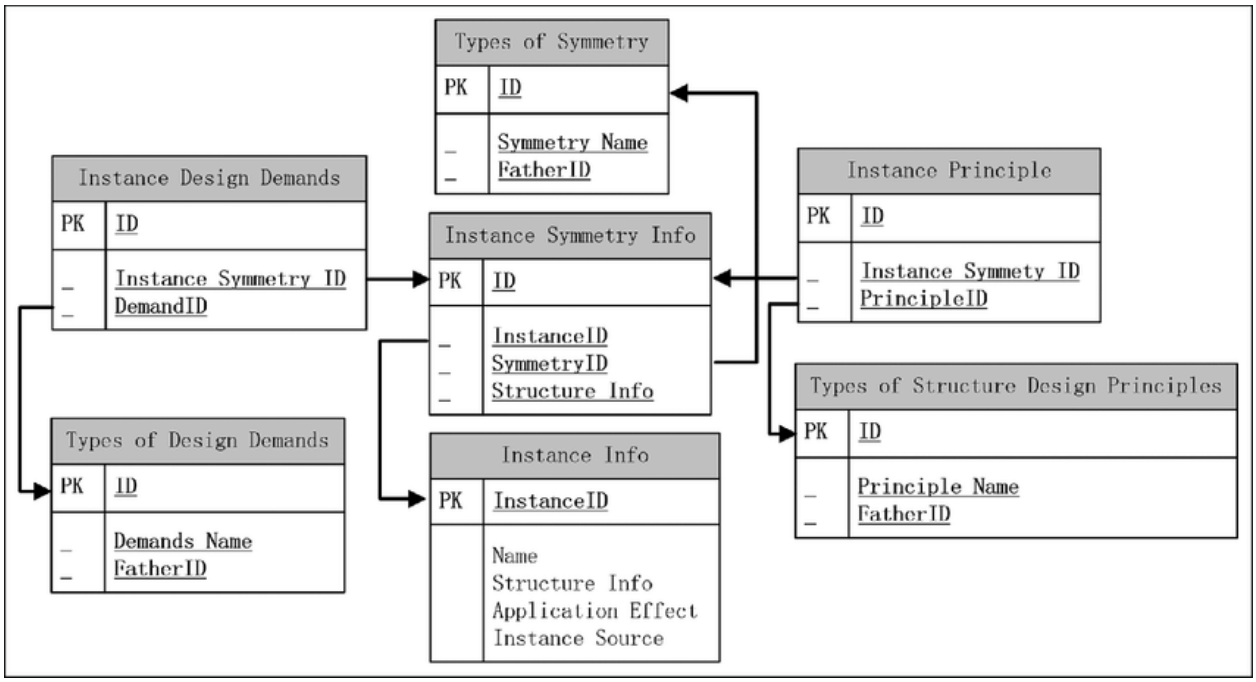

Рисунок 1.1 – Структура фізичної бази даних

Файли даних бувають двох типів:

• Primary File (основний або головний файл);

• Secondary File (вторинний або додатковий файл).

Кожна база даних має один і лише один основний або головний файл (Primary File). Якщо база даних включає лише один файл даних, то цей файл буде основним. Основний файл призначений зберігання всіх системних таблиць, присутніх у будь-якій базі даних. В основному файл зберігається інформація про структуру бази даних, створених в ній об'єктах, параметри

додаткових файлів і файлів журналу транзакцій. За промовчанням основному файлу бази даних надається розширення mdf.

На відміну від основного файлу, база даних може містити безліч вторинних або додаткових файлів (Secondary File) або не містити їх зовсім. У додаткових файлах може зберігатися лише інформація користувача. Зберігання будь-якої системної інформації не допускається. Під час експлуатації бази даних адміністратор може додавати нові або видаляти існуючі додаткові файли.

Файли журналу транзакцій бувають лише одного типу – Transaction Log File, службовця зберігання журналу транзакцій. У базі даних має бути щонайменше один файл журналу транзакцій. Для прискорення обробки транзакцій можна використовувати кілька журналів транзакцій, які розміщені на різних фізичних дисках.

Для кожного файлу бази даних можна встановити властивість автоматичного зростання і крок приросту в мегабайтах або у відсотках від початкового зростання, а також максимальний розмір, до якого можливе зростання файлу. Кожен файл, який використовується у базі даних, має два імені:

• Logical File Name – логічне ім'я файлу, яке використовується у командах Transact-SQL для посилання на конкретний файл;

• OS File Name – ім'я файлу в операційній системі, яке використовується для доступу до файлу в операційній системі.

Складні бази даних можуть мати кілька файлів для даних та журналу транзакцій. І тут файли БД об'єднуються у групи спрощення адміністрування бази даних. SQL Server забезпечує створення груп наступних трьох типів:

• Primary File Group – основна група файлів, яка включає первинний файл та всі файли, не включені до інших груп, база даних може мати лише одну основну групу файлів;

• User-defined File Group – власна група файлів, створювана командою CREATE DATEBASE або командою ALTER DATABASE, якщо в них

використовується параметр FILEGROUP, в базі даних можна створити кілька груп груп файлів з довільним набором файлів;

• Default File Group – це група файлів за замовчуванням, якою призначається одна з груп файлів, створених у базі даних. Лише одна група файлів може бути групою за промовчанням. Якщо не зазначено явно, за замовчуванням стає основна група. Якщо при створенні об'єкта бази даних не зазначено явно, до якої групи файлів він належатиме, цей об'єкт створюється в групі файлів за замовчуванням.

Коли якісь дані записуються у групу файлів, вони розподіляються між файлами цієї групи поступово, тобто. проводиться розпаралелювання запису даних. Для цих цілей можна використовувати і можливості файлової системи NTFS: набір дисків, що чергується, з контролем парності і без нього.

Будь-яка група файлів, у тому числі й основна, може бути встановлена в режимі лише для читання, що дозволяє захистити дані, записані на файли цієї групи.

1.2 LINQ – запити до бази даних мовою C#

Представлений у Visual Studio 2008 і розроблений Андерсом Хейлсбергом, LINQ (Language Integrated Query) дозволяє писати запити навіть без знання мовних запитів, таких як SQL, XML і т.п. Д. Запити LINQ можуть бути написані для різних типів даних.

```
using System;
using System.Linq;

class Program {
   static void Main() {

      string[] words = {"hello", "wonderful", "LINQ", "beautiful", "world"};

      //Get only short words
      var shortWords = from word in words where word.Length <= 5 select word;

      //Print each word out
      foreach (var word in shortWords) {
         Console.WriteLine(word);
      }

      Console.ReadLine();
   }
}
```

Типи LINQ:

- LINQ до об'єктів
- LINQ до XML (XLINQ)
- LINQ до DataSet
- LINQ to SQL (DLINQ)
- LINQ до сутностей

Крім вищезазначеного, існує також тип LINQ з ім’ям PLINQ, який є паралельним LINQ від Microsoft.

LINQ має трьохрівневу архітектуру, в якій найвищий рівень є з мовних розширення, а нижній рівень є джерелом даних, які зазвичай є об'єктами, що реалізують універсальні інтерфейси IEnumerable <T> або IQueryable <T>. Архітектура показана нижче.

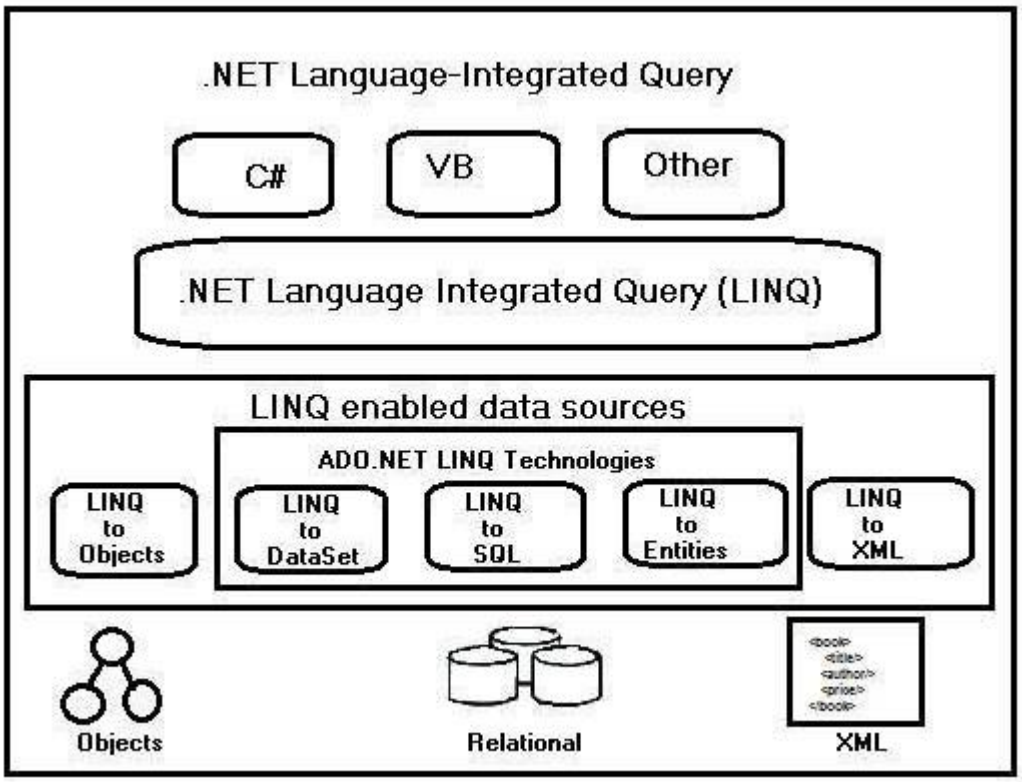

Рисунок 1.2 – Архітектура LINQ в .NET

Вираз запиту – це запит LINQ, виражений у формі, аналогічним запитом SQL, з такими операторами запиту, як Select, Where і OrderBy. Вираз запиту зазвичай починаються з ключового слова «От».

LINQ пропонує безліч переваг і серед них головне – це його потужна виразність, яка дозволяє розробникам виражати декларативно. Деякі з інших переваг LINQ наведено нижче.

- LINQ пропонує підсвітку синтаксису, яка корисна для виявлення помилок під час розробки.
- LINQ пропонує IntelliSense, що означає, що ви можете легко писати більш точні запити.
- Написання кодів у LINQ відбувається набагато швидше, тому час розробки також значно скорочується.
- LINQ прощає відладку завдяки інтеграції з мовою C #.

- З LINQ легко розглядати відносини між двома таблицями завдяки своїй ієрархічній функції, що дозволяє надавати запити, об’єднуючи кілька таблиць за менший час.
- LINQ дозволяє використовувати єдиний синтаксис LINQ при запиті множини різних джерел даних, і це головним чином із-за його універсального походження.
- LINQ є розширеним, що означає, що можна використовувати знання LINQ для запитів нових типів джерел даних.
- LINQ пропонує можливість об'єднати кілька джерел даних на один запит, а також розбити складні проблеми на наборі коротких запитів, які легко відкладати.

1.3 Оптимізація запиту на веб-систему. Розвантаження отримання даних із серверної частини на клієнтську

Перш ніж оптимізувати будь-які запити, потрібно вирішити, які з них найкраще оптимізувати. На жаль, багато людей пропускають цей важливий крок, але, орієнтуючись на конкретні, клопітні запити, які значно впливають на час виконання, можна значно підвищити продуктивність.

Якщо менш вибірково вирішувати, які запити оптимізувати, можна втратити час і гроші, оптимізуючи ті, які не сприяють значному підвищенню продуктивності, не впливають на інші запити або не призводять до проблем, які користувачі помітять.

Запускаючи процес оптимізації запитів до бази даних MS SQL, потрібно запити, які постійно або час від часу повільні, мають червоні прапорці або вносять значний внесок у загальний час виконання.

Запити, які потребують покращення, необхідно оптимізувати. Як згадувалося раніше, зменшення кількості обчислень, які програмне та апаратне забезпечення виконує під час запиту, може призвести до оптимальної

продуктивності SQL-запиту та скорочення часу виконання. Нижче наведено кілька порад і найкращих методів оптимізації запитів до бази даних MS SQL.

Намагаючись знайти конкретні дані, важливо використовувати відповідний запит, і перший крок до пошуку потрібного — це вирішити, які дані треба отримати. Потрібно чітко визначити свої вимоги перед написанням запиту, що дозволить отримувати лише потрібну інформацію, потенційно скоротити час виконання та оптимізувати запити SQL.

Оптимізація таблиць SQL є важливою частиною оптимізації запитів до бази даних MS SQL. Щоб переконатися, що ми отримуємо лише потрібну інформацію, необхідно відфільтрувати свої дані. Фільтрація даних зменшить розмір таблиці та оптимізує час виконання SQL-запитів. Популярні методи оптимізації таблиць SQL та підвищення швидкості запитів включають:

- Надання обмеженого діапазону дат для даних часових рядів
- Обмеження набору даних у підзапиті
- Уникнення дублювання даних
- Спрощення об’єднання

Іноді, коли запит об’єднує таблиці, він різко збільшує кількість рядків у наборі результатів, що може призвести до повільного виконання. Перш ніж об’єднати таблиці, необхідно зменшити їх розмір, як описано вище.

Прості дії, як зміна порядку, у якому об’єднується таблиці, також може оптимізувати запити SQL. При об’єднанні двох таблиць потрібно почати з тієї, яка дасть найменшу кількість результатів після фільтрації.

Використовуючи поля SELECT FROM замість SELECT * FROM, можна звузити дані, отримані з таблиці під час запиту, збільшуючи швидкість запиту. Команда SELECT * витягне всі дані з вашої таблиці, тоді як вказівка полів може скоротити час виконання запиту, забезпечуючи отримання лише необхідних даних.

Аналогічно, використання SELECT ID замість SELECT DISTINCT може зменшити потужність обробки, необхідну для виконання запиту, при цьому повертаючи недубльовані записи. Наприклад, якщо необхідно знайти

конкретного клієнта, запит SELECT DISTINCT John, Smith, Canada FROM Customers може дати кілька результатів, оскільки багатьох людей у Канаді звуть Джон Сміт. SELECT DISTINCT покладається на речення GROUP BY, яке вимагає великої потужності обробки. Щоб додатково відфільтрувати результати та використовувати меншу потужність обробки, використаємо SELECT ID, John, Smith, Canada, Manitoba, Winnipeg, R2C 0A1 FROM Customers.

Хоча можна використовувати як EXIST(), так і COUNT(), щоб дізнатися, чи є в таблиці певний запис, використання EXIST() є ефективнішим. У той час як COUNT() шукатиме всю таблицю, щоб надати загальну кількість відповідних записів, EXIST() працюватиме лише до тих пір, поки не знайде перший запис запису в таблиці, заощаджуючи ваш час і обчислювальну потужність, а також дозволяючи оптимізувати запити SQL.

Іншим методом оптимізації запитів SQL є використання WHERE замість HAVING. Запити WHERE виконуються швидше, ніж запити HAVING. Запити WHERE фільтрують записи перед створенням груп, тоді як запити HAVING фільтрують дані з груп. В результаті використання WHERE замість HAVING є легкою стратегією для оптимізації запитів SQL.

Додавши EXPLAIN на початок запиту, можна переглянути план виконання запиту та отримати краще уявлення про тривалість виконання. Хоча план виконання запиту не завжди точний, він відображатиме як порядок виконання запиту, так і його вартість (вища вартість означає довший час виконання).

Швидше отримувати дані та оптимізувати запити SQL, використовуючи кластеризовані та некластеризовані індекси SQL Server. Індекси можуть скоротити час виконання, але також важливо враховувати, скільки місця на диску вони потребують.

У Microsoft SQL Server не потрібен додатковий дисковий простір для кластерних індексів. Однак, якщо розробники використовують

некластеризовані індекси, ці індекси зберігаються окремо і вимагають більше місця на диску.

1.4 Збережені процедури та їх імплементація

Процедури, що зберігаються, дозволяють підвищити продуктивність, розширюють можливості програмування і підтримують функції безпеки, недоступні при використанні команд Transact-SQL, що відсилаються для обробки на сервер. Підвищується продуктивність - за рахунок локального (стосовно бази даних) зберігання, перекомпіляції вихідного тексту та кешування Можливості програмування розширюються завдяки застосуванню таких поширених засобів програмування, як використання вхідних та вихідних параметрів, а також завдяки багаторазовому використанню процедур. Функції безпеки мають на увазі шифрування тексту процедури та обмеження привілеїв. В результаті ми отримуємо обмежений доступ до внутрішньої структури бази даних, проте нам дозволено запускати збережені процедури, що виконують різні дії над базою даних.

При пересиланні кожної команди (або пакету команд) Transact-SQL на сервер для обробки останній повинен визначити, чи є у відправника права виконання цих команд і чи допустимі самі команди. Перевіривши права доступу та синтаксис команд, SQL Server будує план виконання запиту. Збережені процедури у разі ефективніші. При створенні вони зберігаються в SQL Server, тому при виклику процедури, що зберігається, її вміст відразу ж обробляється сервером. Один - єдиний оператор дозволяє викликати складний сценарій Transact-SQL, який міститься в процедурі, що зберігається, що дозволяє уникнути пересилання через мережу сотень команд.

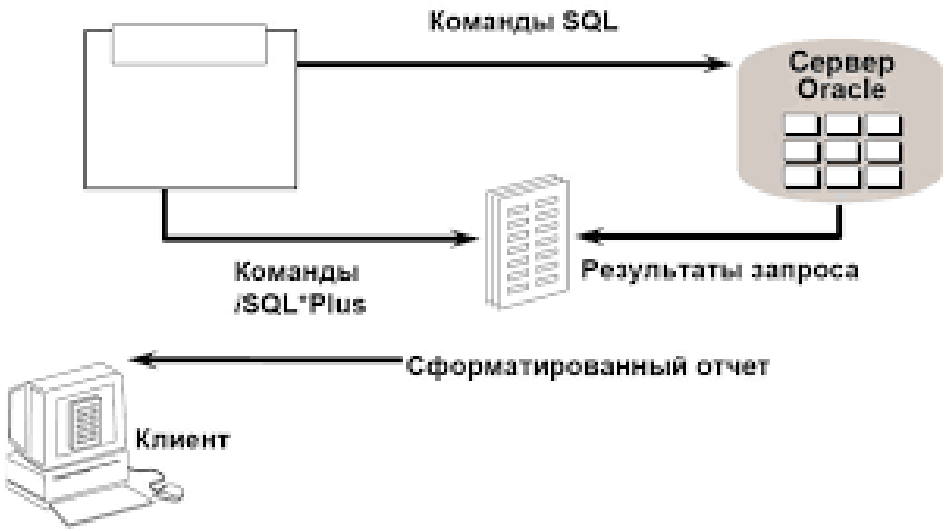

Рисунок 1.2 – Перенаправлення запиту з бази даних на клієнта

Перед створенням процедури, що зберігається, її команди проходять синтаксичну перевірку. Якщо не виявлено жодної помилки, ім'я процедури зберігається у таблиці SysObjects, та її текст - у таблиці SysComments. При першому запуску збереженої процедури створюється план виконання і процедура, що зберігається, компілюється. Надалі її обробка здійснюється швидше, оскільки SQL Server не доводиться перевіряти синтаксис команд, створювати план виконання та компілювати текст процедури. До створення нового плану в кеші перевіряється наявність плану виконання.

Відносний приріст продуктивності, що викликається розміщенням планів виконання процедур, що зберігаються в кеші процедур, зменшується, оскільки плани виконання всіх операторів SQL тепер зберігаються в кеші процедур. При виконанні оператора Transact-SQL по можливості робиться спроба використання існуючого плану виконання.

Створену процедуру, що зберігається, можна викликати в будь-який момент, при виникненні найменшої необхідності. Це забезпечує модульність та стимулює повторне використання коду. Останнє полегшує супровід бази даних, оскільки вона ізольована від мінливих бізнес-правил. Модифікувати процедуру, що зберігається, відповідно до нових правил можна в будь-який момент. Після цього всі програми, що її використовують, автоматично

прийдуть у відповідність до нових бізнес-правил без безпосередньої модифікації.

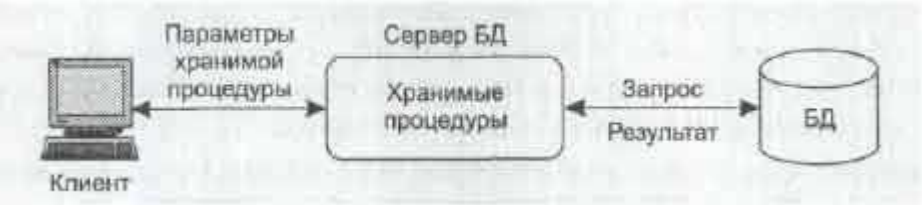

Рисунок 1.3 – Обмін даними між процедурою та сервером БД

Збережені процедури здатні приймати вхідні параметри, повертати значення вихідних параметрів, підтримувати зворотний зв'язок з користувачем через виведення кодів стану та текстових повідомлень, а також викликати інші процедури. Наприклад, одна процедура, що зберігається, може повертати інший код стану, в залежності від якого остання виконує ті або інші дії.

Збережену процедуру пишуть для вирішення якогось одного завдання - в результаті її можна використовувати в декількох базах даних.

Інше важливе призначення процедур, що зберігаються - підвищення безпеки за допомогою ізоляції та шифрування. Користувачам можна надати право на виконання процедури, що зберігається, без безпосереднього доступу до об'єктів бази даних, з якими працює процедура, що зберігається. Крім того, якщо процедуру, що зберігається, зашифрувати при створенні або модифікації, користувачам не вдасться прочитати команди Transact-SQL, що складають процедуру. Ці функції безпеки дозволяють ізолювати від користувача структуру бази даних, що забезпечує цілісність даних та надійність бази.

Існує п'ять класів процедур, що зберігаються: системні, локальні, тимчасові, розширені та віддалені. Є й інші способи класифікації, але це дозволяє легко описати місцезнаходження, призначення і можливості процедури, що зберігається.

• Системні процедури, що зберігаються, знаходяться в базі даних Master. Як правило, їх імена починаються з префіксу sp_. Вони призначені для підтримки функцій SQL Server (зокрема процедур для роботи з каталогом). До

них відноситься вибірка даних із системних таблиць зовнішніми програмами, адміністрування бази даних та управління безпекою.

• Локальні процедури, що зберігаються, зазвичай знаходяться в базі даних користувача. Як правило, їх створюють для вирішення певних завдань у конкретній базі даних. Локальні процедури, що зберігаються, також дозволяють налаштовувати системні збережені процедури.

• Тимчасова процедура, що зберігається, схожа на локальну, проте вона існує лише до закриття з'єднання, в якому створена, або до завершення роботи SQL Server. Залежно від типу, така процедура видаляється після завершення роботи сервера або розриву з'єднання. Непостійність обумовлено тим, що тимчасові процедури, що зберігаються, знаходяться в базі даних TempDB. При кожному запуску сервера ця база створюється знову, тому після закриття сервера всі об'єкти цієї бази даних зникають. Тимчасові процедури, що зберігаються, корисні при роботі з більш ранніми версіями SQL Server, які не підтримують повторне використання планів виконання, а також у тих випадках, коли немає сенсу зберігати процедуру, оскільки значення її параметрів постійно змінюються.

• Розширені процедури, що зберігаються, звертаються до зовнішніх програм, скомпільованих у вигляді 32-розрядних DLL. Деякі системні процедури, що зберігаються, також розглядаються як розширені. Угода про ім'я передбачає використання у іменах розширених збережених процедур префікса xp_. Однак імена деяких розширених процедур починаються з префіксу sp_, а в іменах деяких інших не розширених процедур використовується префікс xp_. Тому не можна розрізнити системні і розширені процедури, що зберігаються, покладаючись лише на відмінності в іменах.

• Як випливає з назви, видалена процедура, що зберігається, працює на віддаленій копії SQL Server. Видалені процедури, що зберігаються, залишені для сумісності з попередніми версіями, в SQL Server 2000 їх замінили розподілені запити.

Існує три типи тимчасових процедур, що зберігаються: локальні (або закриті), глобальні і створювані безпосередньо в TempDB.

Локальна процедура завжди починається із символу #, а глобальна - з ##. При виконанні тимчасової процедури, що зберігається, її область дії обмежена з'єднанням, в якому вона створена. Однак така процедура видима всім користувачам, які встановили з'єднання з базою даних, у вікні Object Browser Query Analyzer. Обмеженість області її дії виключає виникнення конфліктів імен коїться з іншими сполуками, у яких створено тимчасові збережені процедури. Щоб гарантувати унікальність імені тимчасової процедури, що зберігається, SQL Server додає до нього набір символів підкреслення і унікальний номер з'єднання. Привілеї локальної процедури не надаються іншим користувачам. Тимчасова процедура, що зберігається, видаляється з TempDB при закритті з'єднання, в якому вона створена.

Глобальні часові процедури дозволяється виконувати будь-якому з'єднанні. Подібно до тимчасових процедур інших типів, вони створюються в базі даних TempDB, тому мають бути унікальні імена. Право виконання глобальної тимчасової процедури автоматично надається ролі public і може бути змінено.

Глобальні часові процедури так само непостійні, як і локальні. Вони видаляються після закриття з'єднання, в якому створено.

Тимчасові процедури, що зберігаються, які створюються безпосередньо в TempDB. відрізняються від локальних та глобальних процедур наступним:

• їм дозволяється налаштувати права доступу;

• вони зберігаються навіть після завершення з'єднання, в якому створено;

1.5 Розміщення та обробка даних проекту на платформі Microsoft Azure

1.5.1 Завантаження та встановлення процедур на Microsoft Azure

Процедури, що зберігаються, - це відмінний спосіб інкапсуляції коду SQL, який зберігається ближче до даних виділеного пулу SQL. Збережені процедури також допомагають розробникам досягти модульності своїх рішень, інкапсулюючи код в керовані одиниці та розширюючи можливості для його повторного використання. Кожна процедура може також приймати параметри, що робить їх ще більш гнучкими.

Виділений пул SQL підтримує спрощену та оптимізовану реалізацію збережених процедур. Головна відмінність від SQL Server в тому, що процедура, що зберігається, не є попередньо скомпільованим кодом.

Зазвичай у сховищах даних час компіляції мало в порівнянні з часом, який потрібний для виконання запитів до великих обсягів даних. Більш важливо переконатися, що код процедури, що зберігається, правильно оптимізований для великих запитів.

Мета - заощадити години, хвилини та секунди, але не мілісекунди. Тому зручніше вважати процедури, що зберігаються контейнерами для логіки SQL.

Коли виділений пул SQL виконує процедуру, що зберігається, інструкції SQL аналізуються, транслюються і оптимізуються. Під час цього процесу кожна інструкція перетворюється на розподілені запити. Код SQL, який виконується з даними, відрізняється від запиту, що надсилається.

Коли процедури, що зберігаються, викликають інші процедури, що зберігаються, або виконують динамічний код SQL, то внутрішню збережену процедуру або код виклику називають вкладеними.

Виділений пул SQL підтримує до восьми рівнів вкладеності. При цьому рівень вкладеності у SQL Server - 32.

Виклик зберігання процедури верхнього рівня вважається 1 рівнем вкладеності.

SQL

EXEC prc_nesting

Якщо процедура, що зберігається, також виконує ще один виклик EXEC, рівень вкладеності збільшиться до 2.

SQL

```
CREATE PROCEDURE prc_nesting
AS
EXEC prc_nesting_2 -- Цей call is nest level 2
GO
EXEC prc_nesting
```

Якщо друга процедура виконує якийсь динамічний SQL-запит, рівень вкладеності збільшиться до 3.

SQL

```
CREATE PROCEDURE prc_nesting_2
AS
EXEC sp_executesql 'SELECT 'another nest level' -- This call is nest level 2
GO
EXEC prc_nesting
```

Виділений пул SQL в даний час не підтримує конструкцію @@NESTLEVEL. Отже, слід відстежувати рівень вкладеності. Перевищення обмеження на вісім рівнів вкладеності малоймовірне. Однак у цьому випадку потрібно переписати код, щоб він відповідав рівням вкладеності у цих межах.

INSERT..EXECUTE

Виділений пул SQL не дозволяє використовувати результуючий набір процедури, що зберігається в інструкції INSERT. Однак є інший підхід, яким можна скористатися.

1.5.2 Розбір завантажуваних процедур та їх порівняння з LINQ – операцією

Структура тесту полягала в налаштуванні статичного методу для повернення даних із таблиці Customers Northwind, придатного для прив’язки

до ObjectDataSource в ASP.NET. Ми виконали два набори тестів, один для повернення шести стовпців з усіх рядків, а другий, щоб повернути ті самі шість стовпців з одного рядка. Кожен набір містив такі варіації:

- DataReader, щоб забезпечити базову продуктивність для порівняння з іншими технологіями.
- DataTable, використовуючи класичні інструменти ADO.NET (DataAdapter запускає команду для заповнення таблиці).
- LINQ to SQL, використовуючи скомпільований запит і з вимкненим відстеженням об'єктів, щоб максимізувати продуктивність. Список результатів проектується безпосередньо із запиту.
- LINQ to Entity Framework, використовуючи скомпільований запит для максимальної продуктивності. Як і у випадку з LINQ to SQL, список результатів проектується безпосередньо із запиту.
- Entity SQL, як альтернатива LINQ, запитує Entity Framework. Структура коду для Entity SQL використовує зчитувач, подібно до використання DataReader з T-SQL.

Як для LINQ to SQL, так і для Entity Framework я використовував інструменти візуального дизайнера, щоб включити в модель лише таблицю Customers.

Тест вимірював час, що минув, і загальний час процесора. Можна припустити, що різниця включає час, використаний SQL Server, а також будь-який інший час поза процесом. Я проводив тести на Dell Latitude E6500 з Vista Ultimate, SQL Server 2008, Intel Core 2 Duo P9500 (2,5 ГГц), 4 ГБ оперативної пам'яті та диску 7200 об/хв. Система не працювала, за винятком тестів; тестові запуски були досить послідовними за часами, виміряними стандартними відхиленнями протягом набору з 10 тестових запусків.

Програма тестування виконувала кожен запит один раз, щоб переконатися, що весь код був завантажений і JIT, а всі плани доступу та дані були кешовані, так що час запуску було виключено для кожного сценарію. Потім програма виконала 10 000 запитів і зібрала загальну інформацію про час

і робочий набір. Для кожного сценарію програму тестування запускали один раз, а потім запускали 10 разів, щоб записати дані часу.

Тест був розроблений для вимірювання лише виконання коду для запитів. Немає бізнес-логіки, і дизайн тесту гарантував, що початкові витрати були виключені з результатів тестування.

Як і очікувалося, використання DataReader із необробленим T-SQL є найкращою технологією та найкращою технологією для надзвичайно великих обсягів даних і для програм, де продуктивність є єдиним, що має значення. DataReader використав 0,40 мілісекунди (минув), щоб отримати 92 рядки та зберегти дані у списку, і лише 0,15 мілісекунди для одного рядка.

DataTable з класичним ADO.NET працював майже так само добре, використовуючи 0,58 мілісекунди (пройшло) для 92 рядків і 0,18 мілісекунди для одного рядка. У наведеній вище діаграмі DataReader використовується як базова лінія для порівняння, тому відносна вартість використання DataTable і DataAdapter становила 1,4 для 92 рядків і 1,2 для одного рядка. Це не так багато накладних витрат в обмін на використання стандартизованої структури, яка включає метадані про імена та типи даних. Використання пам’яті було практично ідентично використанню пам’яті для DataReader.

LINQ to SQL також працював дуже добре, використовуючи 0,63 мілісекунди (пройшло) для 92 рядків і 0,36 мілісекунди для одного рядка. Коефіцієнт продуктивності порівняно з DataReader становить 1,6 для 92 рядків і 2,3 для одного рядка. Порівняно з DataTable, коефіцієнт продуктивності (без діаграми) становив 1,2 для 92 рядків і 1,9 для одного рядка. LINQ to SQL використовував 40 МБ додаткової пам’яті на основі кінцевого розміру робочого набору в кінці кожного запуску.

Це дуже пристойна продуктивність, враховуючи додаткові витрати, хоча Ріко Маріані з Microsoft отримав ще кращі показники. У наших тестах усі запити встановлювали нові об’єкти з’єднання (або контексти даних) для кожного запиту.

З Entity Framework є значні додаткові витрати на продуктивність. LINQ to EF використав 2,73 мілісекунди (минув) для отримання 92 рядків і 2,43 мілісекунди для одного рядка. Для 92 рядків це коефіцієнт продуктивності 6,8 порівняно з DataReader, 4,7 порівняно з DataTable та 4,4 порівняно з LINQ to SQL (останні два не наведені вище). Для одного рядка LINQ to EF використовував 2,43 мілісекунди (пройшло), з коефіцієнтом продуктивності 16,0 порівняно з DataReader, 13,2 порівняно з DataTable та 6,8 порівняно з LINQ to SQL. Використання пам’яті для LINQ to EF було приблизно на 130 МБ більше, ніж для DataReader.

Запити SQL до EF виконували приблизно так само, як і LINQ до EF, з 2,78 мілісекунд (минув) для 92 рядків і 2,32 мілісекунди для одного рядка. Використання пам'яті було подібним до LINQ to EF.

1.5.3 Огляд оптимізації «перфомансу» веб-сторінки

Оптимізація перфомансу проводилася у 2 етапи:

1. Тестування запиту щодо його вплив на систему.

Наприклад, ми вибираємо запит на отримання даних у GET запиті, який у свою чергу отримує список даних із серверної частини, для подальшої передачі його рендеригу в клієнтській частині. Для початку подивимося як виглядає середній час запиту на отримання даних без процедур, що зберігаються:

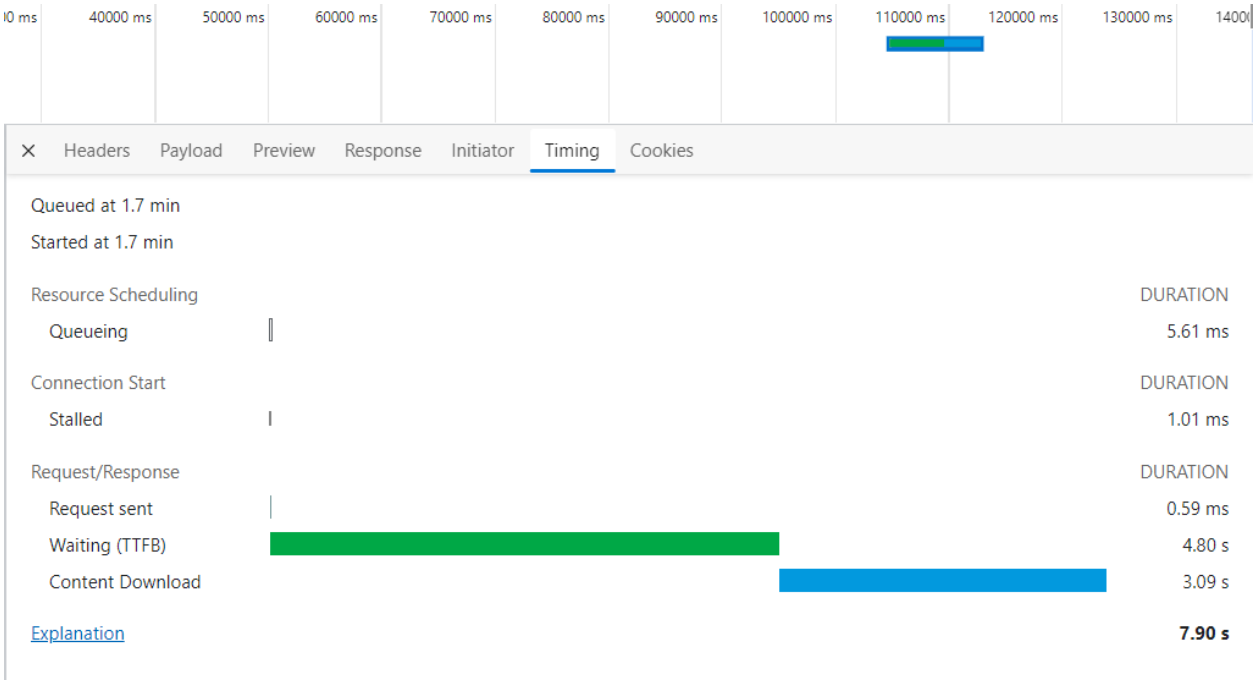

Рисунок 1.5 – Час виконання запиту на отримання списку "Activities" даного власника

Як видно з малюнка, час виконання запиту на невеликий проміжок даних у списку час запиту становить 7.90 секунд, що істотно довго для очікування користувача, при цьому час на рендеринг даних виявилося 3.09 секунд, що так само істотно довго. Основною проблемою в ході підвантаження даних виявилося, те, що даний тест виконувався з використанням вайфай підключення до супершвидкісного інтернету, якщо цей тест провести із затримкою підключення (наприклад, як показано на малюнку нижче, можна встановити «тротлінг» підключення на Slow 3G), то отримані дані ускладнюватимуть час очікування відповіді від сервера, що є неприпустимим для кінцевого користувача продукту.

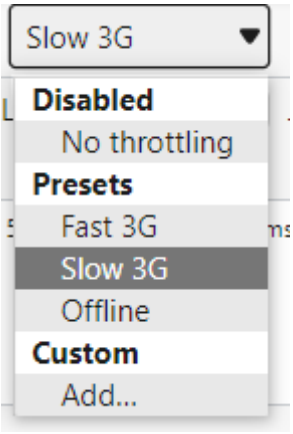

Рисунок 1.6 – Приклад списку тролінга підключення в браузері Microsoft Edge

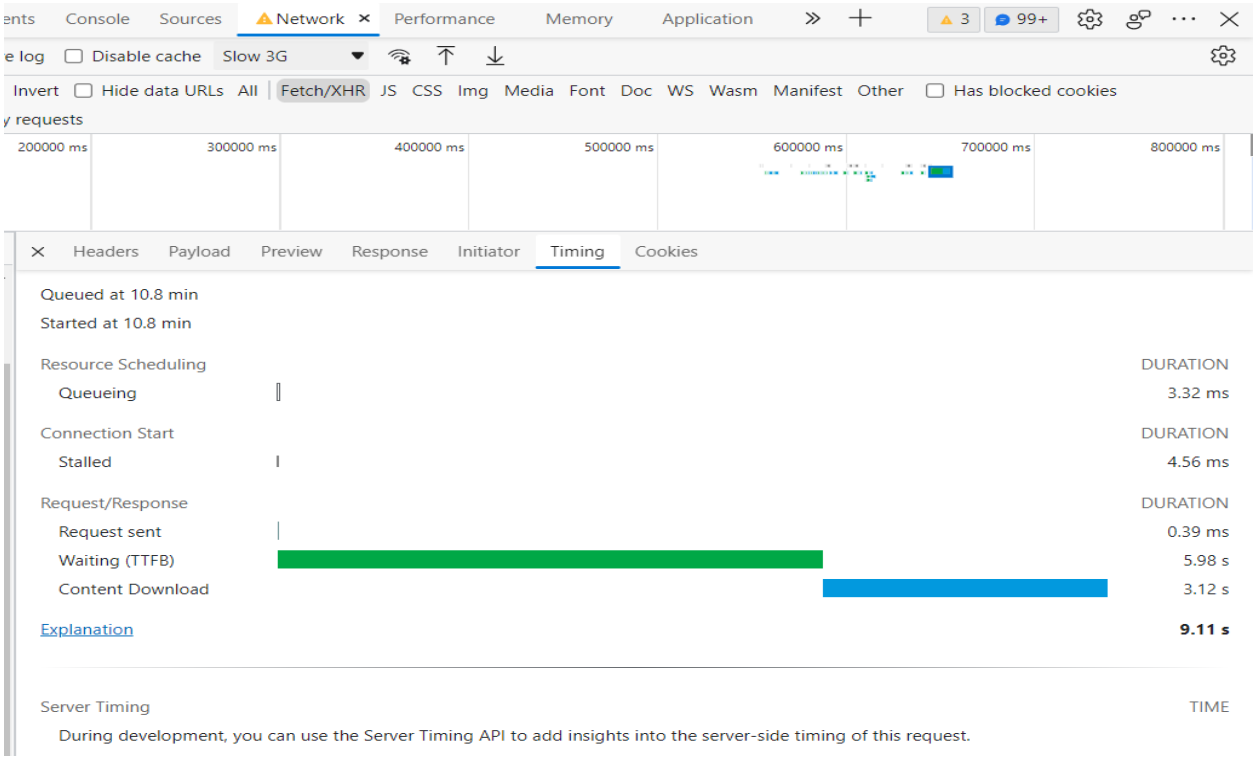

Рисунок 1.7 – Приклад отримання запиту з використанням тротлінгу

Як видно з відповіді від сервера, час отриманої відповіді склало 9.11 секкунд з яких, так само довелося очікувати 3.11 секкунд відмальовування результату отримання даних від клієнтської частини.

Отже, приходимо до висновку, що Waiting запит від сервера грає велику роль в отриманні даних.

Виходячи з отриманих методів і поєднавши їх в 1 API запит, ми прибрали структуру LINQ - запиту і замінили його на запит процедури, що зберігається, тим самим оптимізувавши. Результати показують, що клієнтська частина рендерит швидше дані, так як сервер надсилає їх вже шляхом не обробки запиту, а шляхом обробки процедури, що зберігається.

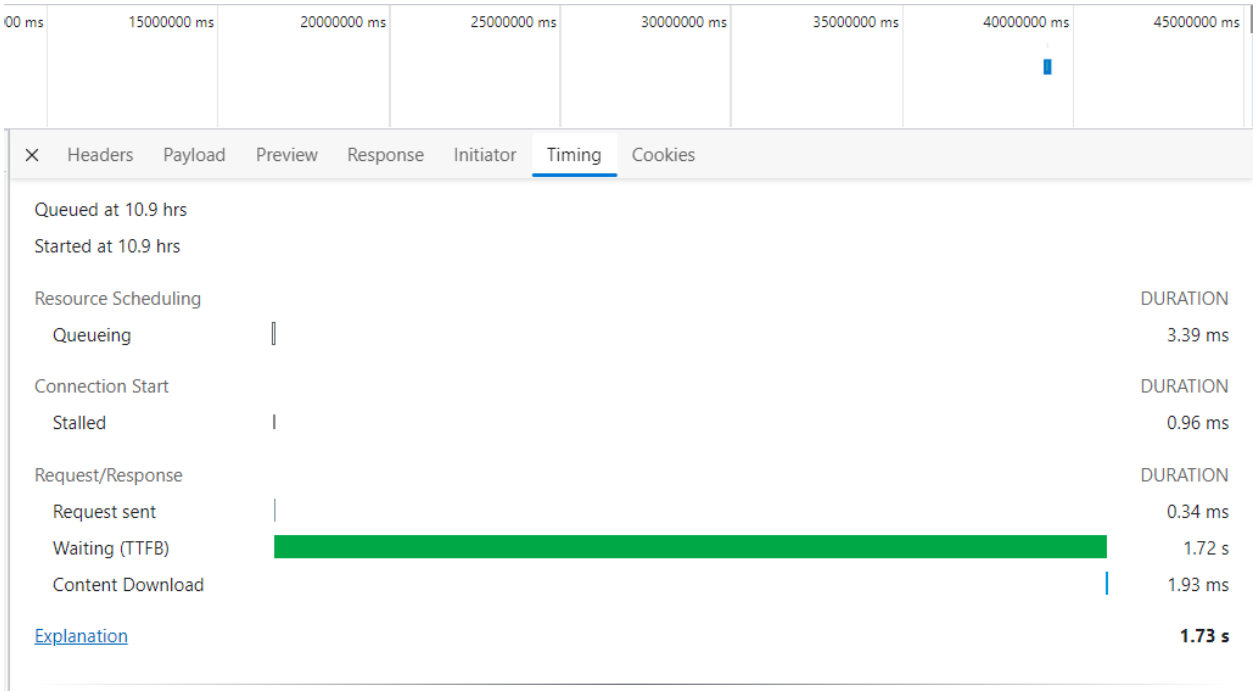

Рисунок 1.8 – Запит, оптимізований збереженою процедурою

Як видно з запиту результати оптимізації «перфофмансу» значно краще показують себе, час завантаження даних для рендерингу склало 1.73 секкунди, замість 7.90, що майже в 6 разів швидше, але найцікавіше, що час рендеригу склало 1.93 мілісеккунди, що значно оптимізації. тільки очікування, але і продуктивність клієнтської частини.

Перевіримо наші результати, додавши тротлінг запиту на повільний інтернет, як показано раніше.

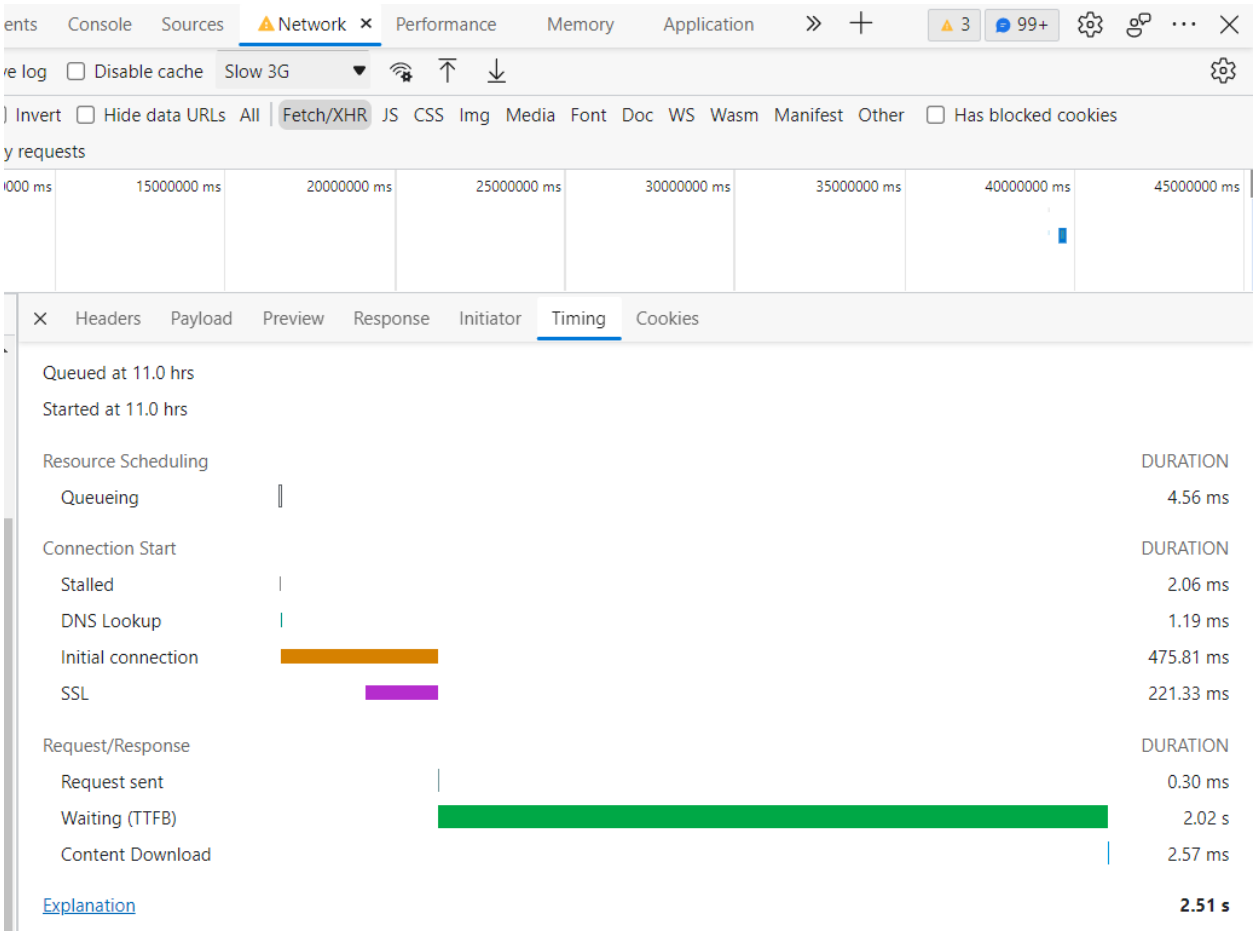

Рисунок 1.9 – Додавання тротлінгу в запит, що надходить.

Як видно з отриманих даних, додавання троттлінга не сильно вплинуло на отримані результати.

2 ЕКОНОМІЧНИЙ РОЗРАХУНОК

2.1 Резюме

Тема даного дипломного проекту «Побудова і оптимізація процедур з бази даних і розміщення їх на хмарі за рахунок інструментів Azure і команд в Kudu.» В роботі були показані методи імплементації та з'єднання даних збережених процедур, а також їх розміщення у хмарі Microsoft Azure.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2. Визначення трудомісткості розробки програмного забезпечення.

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

Каталог аналогів

Таблиця 4.1

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП автоматизованих розрахунків	1300 – 8600
2. ПП загальної математики і ПП імітаційного моделювання	1800 – 8800
3. ПП введення інформації	1060 – 5750

У таблиці 4.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.4.2

Таблиця.4.2

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): T^a = 244 х 0,7 = 170,08 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (4.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (4.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (4.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 4.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 4.3.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 4.4.).

Таблиця 4.2.Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 4.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 4.4. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Т^а *L_1*К_н= 170,08*0,12*0,7= 14,35 \quad (люд/годин) \qquad (4.1)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Т^а *L_2*К_н =170,08*0,11*0,7 = 13,10 \quad (люд/годин) \qquad (4.2)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Т^а *L_3*К_н*К_т=170,08*0,61*0,7*0,7= 50,81 \quad (люд/годин) \qquad (4.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стр), розробка ТП $N_{тп}$=12 (стр), розробка робочого проекту $N_{рп}$=14(стр), пояснювальна записка відповідно $N_{пз}$=22(стр) Розрахунок зведений у таблицю 4.5

Таблиця 4.5. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Ртз}$=14,35	$T_{кк}$=0,7*$N_{тз=}$ 0,7*2=1,4	$T_{нк}$=0,15*$N_{тз}$=0,15*2= 0,3
2.Розробка ТП	$T_{Ртп}$=13,10	$T_{кк}$=0,7*$N_{тп}$=0,7*12=8,4	$T_{нк}$=0,15*$N_{тп}$=0,15*12=1,8
3.Розробка РП	$T_{Ррп}$=50,81	$T_{кк}$=0,7*$N_{рп}$=0.7*14=9,8	$T_{нк}$=0,15*$N_{рп}$=0,15* 14=2,1
4.Розробка ПЗ	$T_{пз}$=1,5*22 *$N_{пз}$=33	$T_{кк}$=0,7*$N_{тз}$=0,7*22=15,4	$T_{нк}$=0,15*$N_{пз}$=0,15*22=3,3
Усього, в т.ч.:	ΣT=153,76		
- на розробку	$ΣT_р$=111,26		
- контроль керівника		$ΣT_{кк}$=35	
- нормоконтроль			$ΣT_{нк}$=7,5

2.3 Розрахунок ціни програмного продукту.

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 4.6. Відповідно до статті 8 «Закону про Державний бюджет України на 2022» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2022 року - 6500 гривень; мінімальну погодинну тарифну ставку – 39.26 грн.

Таблиця 4.6 Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	111,26	45,00	5006,25

2.Контроль керівника	35	50,00	1750
3.Нормоконт-роль	7,5	50.00	375
Усього	-	-	ΣЗо= 7131,25

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 4.7

Таблиця 4.7 Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	50	2,5	125,00
Транспортно– заготівельні витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$= 125*0,1= 12,5
Усього				$В_м$=$В_{мі}$+$В_{тр\ з=}$ 137,5

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 4.8.

Таблиця 4.8. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	137,5	$В_м$ = 137,5
2. Основна заробітна плата	7131,25	$З_о$ = 7131,25
3.Додаткова заробітна плата	1069,69	$Зд = 0{,}15 \times Зо$ =0,15*7131,25
4.Відрахування до єдиного фонду соціального внеску	1804,21	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22*(7131,25 + 1069,69)
5. Накладні витрати	2139,38	$Внак. = 0{,}3 \times Зо$ =0,3 * 7131,25
6. Повна собівартість	12282,02	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.=

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_п*Р)/100=(12282{,}02*10)=1228{,}20$ (4.4)

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о=С_{пов}+П=12282{,}02+1228{,}20=13510{,}22$ (4.5)

Податок на додану вартість визначаємо по наступній формулі:

$ПДВ = 0.2 * Ц_о$ =0,2*13510,22= 2702,04 (4.6)

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$$Ц_р = Ц_о + ПДВ = 13510,22 + 2702,04 = 16212,26 \quad (4.7)$$

3 ОХОРОНА ПРАЦІ

Поліпшення умов та охорона праці стає одним з важливих напрямків матеріального та культурного рівня життя народу, а це, у свою чергу, сприяє зростанню якості та продуктивності праці, підвищенню соціально-економічних показників виробництва, зменшенню коштів на витрати від травматизму, професійних захворювань і аварій.

В дипломному розділі дипломного проекту розглядається питання охорони праці програміста на стадії побудови ы оптимізації процедур з бази даних і розміщення їх на хмарі за рахунок інструментів Azure і команд в Kudu

1 Аналіз небезпечних та шкідливих чинників, що впливають на працівника.

На основі аналіз умов праці, технологічних процесів, апаратури і обладнання з точки зору можливості виникнення появи небезпечних факторів, виділення шкідливих виробничих речовин визначаються небезпечні ділянки виробництва, можливі аварійні ситуації, розробляються заходи щодо їх усунення або обмеження наслідків.

На операторів ПК і програмістів мають вплив таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, недостатня освітленість робочої зони, електричний струм та інші. Тому на робочому місці програміста повинні бути створені умови для високопродуктивної праці.

Робота може кваліфікуватися як робота програміста (оператора) ЕОМ.

2 Розробка заходів з охорони праці

Вибір технічних засобів забезпечення безпеки повинен здійснюватися на основі вивчення особливостей кожного виявленого небезпечного й шкідливого виробничого фактора і зони його дії – так званої небезпечної зони.

2.1 Виробничі приміщення

У приміщеннях , призначених для роботи з , доцільно, щоб вікна були орієнтовані на північ або північний захід. На вікнах повинні бути штора або жалюзі, що регулюють рівень освітленості і захищають від прямого влучення сонячних променів на робоче місце.

При кольоровому оформленні виробничих і допоміжних приміщень необхідно враховувати орієнтацію їхніх вікон стосовно частин світла і використовувати гармонійне сполучення кольорів. Використовують мало насичені (основні) кольори, для невеликих помешкань або ділянок, що рідко потрапляють у поле зору працюючих, а також для створення контрастності - кольори середньої насиченості (допоміжні), для маленьких по площі поверхонь - насичені (акцентні) - як функціональне фарбування. Стелі у всіх приміщеннях повинні бути білими.

2.2 Мікроклімат робочої зони працівників вентиляція.

Робота програміста за енерговитратами відноситься до категорії легких робіт Іа, Іб, тому повинні дотримуватися наступні вимоги згідно ДСН 3.3.6.042-99:

оптимальна температура повітря – 22°С (допустима – 20-24°С);

оптимальна відносна вологість – 40-60% (допустима – не більш 75%);

швидкість руху повітря не більш 0,1 м/с.

Виміряні за допомогою приладів (психрометр Августа) температура та вологість у приміщенні відповідають для теплого періоду року.

2.3 Вентиляція приміщення.

Для створення в приміщенні нормальних умов мікроклімату для працівника і видалення шкідливих забруднень, була спроектована і належним чином встановлена вентиляційна система – загально обмінна, припливно-витяжна по нормам ДСТУ Б А. 3.2-12:2009 ССБП.

Вентиляція створює на робочому місці, метеорологічні умови і чистоту повітряного середовища, що відповідаюсь чинним санітарним нормам. Разом з тим вентиляція забезпечує умови, що відповідають вимогам технологічного процессу.

2.4 Освітлення робочого місця, шум, вібрація

Приміщення, в яких встановлені персональні комп'ютери, повинні мати природне та штучне освітлення. Природне освітлення здійснюється через світові прорізи (вікна), орієнтовані переважно на північ чи північний схід. Штучне освітлення в приміщені здійснюється системою загального рівномірного освітлення. На поверхні столу в зоні розміщення документів штучне освітлення має становити 300-500лк.

2.5 Шум

Так як шум має 35Дб, сприйняття шуму людським вухом межуется від 20Дб до 120 дб, це означає, що при роботі за ЕОМ шум не заважає, працівнику працювати.

Для запобігання виникнення інших шумів у відповідності з ГОСТ 12.1.029-80 зниження шуму й вібрації в приміщенні дипломним проектом передбаченні звукоізоляція вікон та дверей.

2.6 Організація робочого місця користувача ПК

Обладнання і організація робочого місця з ВДТ мають забезпечувати відповідність конструкції всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності (ГОСТ 12.2.032-78, ГОСТ 22.269-76, ГОСТ 21.889-76).

Робочі місця слід так розташовувати відносно світових прорізів, щоб природне світло падало збоку, переважно зліва. При розміщенні робочих столів з ВДТ слід дотримуватися таких відстаней: між бічними поверхнями ВДТ -1,2м; від тильної поверхні одного ВДТ до екрану іншого – 2,5м. Екран ВДТ має розташовуватися на оптимальній відстані від очей користувача, що становить 600…700 мм, але не ближче ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів.

Клавіатуру розташовують на поверхні столу на відстані 100…300 мм від краю, зверненого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій, який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5…15^0.

При оснащенні робочого місця лазерним принтером параметри лазерного випромінювання повинні відповідати вимогам СанПіН № 5804-91.

5.7 Вимоги до режимів праці та відпочинку при роботі з ЕОМ

Збереження високої продуктивності праці користувачів ВДТ може бути досягнуто методами запровадження раціонального режиму праці та відпочинку.

Впровадження режимів праці та відпочинку повинна передувати робота щодо наукового обґрунтування тривалості та порядку проведення перерв, заснована на урахуванні змісту праці та факторів, які обумовлюють умови праці.

Правилами встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці:

для розробників програм із застосуванням ЕОМ слід призначати

регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну

годину роботи за ВДТ;

для операторів із застосуванням ЕОМ слід призначати регламентовані перервидля тривалістю 10 хвилин після кожної години роботи за ВДТ.У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані відпочинку тривалістю 15 хвилин через кожні дві години;

для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години.

Конструювання раціонального режиму праці та відпочинку залежить від рівня спеціальних знань користувача при застосуванні конкретного програмного забезпечення, а отже, від його здатності раціонально вирішувати нові завдання. Тому планування режиму робіт слід проводити, враховуючи кваліфікацію користувача, який вирішує конкретне завдання.

Для надійної роботи користувача у системі потрібно мати короткі довідники, які повинні бути на кожному робочому місці.

При використанні ВДТ програмістами та іншими професійними групами, для яких специфічним є творчий компонент при виконанні виробничих завдань, конструювання раціонального режиму праці доцільно здійснювати на основі врахування індивідуальних стратегій діяльності та особливостей стану здоров'я користувачів

4 Пожежна безпека

Забезпечення пожежної безпеки об'єкта здійснюють на основі вимог, а також інших нормативних документів .

Установлюють небезпечні фактори пожежі, вплив яких призведе до травми чи загибелі людей, а також до матеріальних

Протипожежний захист приміщення забезпечується застосуванням автоматичної установки пожежної сигналізації, наявністю засобів пожежегасіння, застосуванням основних будівельних конструкцій будинку з регламентованими межами вогнестійкості, організацією своєчасної евакуації людей.

Основними причинами пожежі є: необережне поводження з вогнем, незадовільний стан електротехнічних установок і невиконання правил їх експлуатації, несправність виробничого обладнання і порушення режимів технологічних процесів, порушення правил пожежної безпеки.

До засобів гасіння пожежі відносяться внутрішні пожежні водопроводи (крани –ПК), вогнегасники (вуглекислотні та порошкові), сухий пісок тощо.

В будівлях пожежні крани встановлюють в коридорах, на майданчиках сходових кліток. Кожний пожежний кран укомплектований пожежним рукавом і розміщений у відповідних ящиках, які знаходяться на висоті 1.35 м від полу.

Для гасіння пожеж на початкових стадіях широко застосовуються вогнегасники. У виробничих приміщеннях це головним чином вуглекислотні вогнегасники, достоїнством яких є висока ефективність гасіння пожежі,

збереження електричного устаткування. Розташовують вогнегасники на видних місцях, на висоті не більше як 1,5 м від полу.

Будівлі укомплектовані пожежними щитами з набором інструментів, біля щитів – бочки з водою, ящики з піском.

Пожежна безпека приміщень, що мають електричні мережі, регламентується ГОСТ 12.1.033-81, ГОСТ 12.1.004-85. Робота оператора ЕОМ повинна вестися в приміщенні, що відповідає категорії Д пожежної безпеки (негорючі речовини й матеріали в холодному стані).

Виробничі приміщення мають запасні виходи. Двері повинні мати освітлений надпис « Запасний вихід». План евакуації вивішується на видному місці у основного виходу із приміщення Виробничі приміщення мають запасні виходи. Двері повинні мати освітлений надпис « Запасний вихід». План евакуації вивішується на видному місці у основного виходу із приміщення

У випадку виникнення пожежі необхідно відключити електроживлення, викликати по телефону 101 пожежну команду, евакуювати людей із приміщення відповідно до плану евакуації і приступити до ліквідації пожежі.

ВИСНОВКИ

Напрямок ООБД виник порівняно давно. Публікації з'явилися вже в середині 80-х років. Проте найактивніше цей напрямок розвивається останніми роками. З кожним роком збільшується кількість публікацій та реалізованих комерційних та експериментальних систем.

Виникнення напряму ООБД визначається насамперед потребами практики: необхідністю розробки складних інформаційних прикладних систем, котрим технологія попередніх систем БД була цілком задовільною.

Найважливішим новим якістю ООБД, якого дозволяє досягти об'єктно-орієнтований підхід, є аспект поведінки об'єктів. У прикладних інформаційних системах, що ґрунтувалися на БД із традиційною організацією, існував принциповий розрив між структурною та поведінковою частинами. Структурна частина системи підтримувалась усім апаратом БД, її можна було моделювати, верифікувати тощо, а поведінкова частина створювалася ізольовано. У середовищі ООБД проектування, розробка та супровід прикладної системи стає процесом, у якому інтегруються структурний та поведінковий аспекти. Звичайно, для цього потрібні спеціальні мови, що дозволяють визначати об'єкти та створювати на їх основі прикладну систему.

Нині ведеться дуже багато експериментальних та виробничих робіт у галузі СУБД. Вже кілька років тому відзначалося існування щонайменше тринадцяти комерційно доступних систем ООБД. Серед них системи O2, ORION, GemStone та Iris.

У цій роботі були показані методи імплементації та з'єднання даних збережених процедур, а також їх розміщення у хмарі Microsoft Azure.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пауэлл Т.А. Полное руководство по HTML.- Мн.: Попурри, – 2001. – 912с.

2. Jaworsky J. Mastering JavaScript and Jscript. – New York: Изд-во Sybex, 2000. – 940с.

3. Николенко Д.В. Практические занятия по Java Script.- М.: Наука и техника, 2000. – 128с.

4. Вагнер Р., Вайк А. Java Script.- К.: ДиаСофт, – 2001. – 464с.

5. Брандебау Дж. Java Script – Сборник рецептов.- СПб.: Питер, – 2000. – 416с.

6. Вуд Л. Web-графика. – СПб: «Диалектика», 2001. –488с.

7. Мюррей Д., Ван Райпер У. Энциклопедия графических форматов. – К.: BHV, 1997. – 672с.

8. Кирсанов Д. Web-дизайн.- СПб.: Символ-Плюс, – 2000. – 376с.

9. Буковецкая О.А. Дизайн текста: шрифт, эффекты, цвет. – М.: ДМК, 2000. – 304с.

10. Вин Дж. Искусство Web-Дизайна. - СПб: Изд-во «Питер», 2002. – 360с.

12. Шапошников И.В. Интернет программирование.- СПб.: BHV, – 2000. – 224с.

13. Федорчук А. Как создаются Web-сайты.- СПб.: Питер, – 2000. – 224с.

14. Леонтьев Б.К. Web-дизайн: тонкости, хитрости, секреты. – М.: Майор, 2001. – 176с.

11. Which is Best for Web Application Development—Dot Net, PHP, Python, Ruby, or Java: [Электронный ресурс]. URL: https://www.addonsolutions.com/blog/which-is-best-for-web-application-development-dot-net-php-python-ruby-or-java.html/ (Дата обращения 15.04.2018);