Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 2

з дисципліни

“ Програмні додатки з використанням баз даних”

по темі

«Наповнення бази даних. Застосування індексів»

Варіант 4

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Cт.вик. Клименко О. М. |  | студент 3-го курсу  гр. ІП-51 ФІОТ  Гладишко  Валентин  Юрійович |

Київ – 2017

**ЗМІСТ:**

[1 Завдання 3](#_Toc496644755)

[2 Виконання Завдання 4](#_Toc496644756)

[3. ВИСНОВОК 20](#_Toc496644757)

# Завдання

**Завдання 1**:

Кожен студент згідно предметної області за варіантом повинен вибрати

таблицю, що містить такі поля як прізвище, імя та по-батькові:

1) Написати програмний додаток (бажано веб-додаток), що має:

а) кнопку «Очистити таблицю»;

б) поле «Ввести кількість рядків» (кількість рядків може бути 500 тис. і

більше);

в) кнопку «Наповнити таблицю». Вивести час, за який відбулося

наповнення вибранної таблиці заданною у пункті б) кількістю рядків. При

наповненні таблиці, звертаю увагу на те, що треба наповнити ще одну

таблицю, що пов’язана з цією для виконання запиту з JOIN.

2) Виконати без індекса запити з WHERE, JOIN та вивести час виконання

кожного запиту.

3) Створити індекси, які на вашу думку дадуть прискорення виконання

попередніх запитів з WHERE, JOIN та вивести час виконання кожного

запиту.

**Завдання 2** (повторення теми «Індекси» - матеріалу попереднього семестру):

1) Описати які типи індексів використовує вибрана вами СУБД і як кожен

тип реалізований алгоритмічно (наприклад, B-tree індекс та принцип його

роботи).

2) Для яких запитів який тип індексу використовує вибрана вами СУБД,

навести приклади, до кожного типу індексу хоча один запит (наприклад,

B-tree індекс використовується, якщо до індексованого поля

застосовуються оператори порівняння). Приклади виключно з вашого

варіанту.

**Преметна область**: Клієнти Стоматологічної Клініки

# Виконання Завдання

1. Завдання по написанню запитів

*“Program.cs”*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace lab2

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

*“Form1.Designer.cs”*

namespace lab2

{

partial class Form1

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

private void InitializeComponent()

{

this.buttonClearTable = new System.Windows.Forms.Button();

this.buttonFillTable = new System.Windows.Forms.Button();

this.CountRows = new System.Windows.Forms.RichTextBox();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.timeFill = new System.Windows.Forms.RichTextBox();

this.label3 = new System.Windows.Forms.Label();

this.timeExecute = new System.Windows.Forms.RichTextBox();

this.buttonExecuteQuery = new System.Windows.Forms.Button();

this.buttonCreateIndex = new System.Windows.Forms.Button();

this.buttonDeleteIndex = new System.Windows.Forms.Button();

this.label4 = new System.Windows.Forms.Label();

this.timeClear = new System.Windows.Forms.RichTextBox();

this.timeCreate = new System.Windows.Forms.RichTextBox();

this.label5 = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// buttonClearTable

//

this.buttonClearTable.Location = new System.Drawing.Point(56, 35);

this.buttonClearTable.Name = "buttonClearTable";

this.buttonClearTable.Size = new System.Drawing.Size(121, 42);

this.buttonClearTable.TabIndex = 0;

this.buttonClearTable.Text = "Очистити таблицю";

this.buttonClearTable.UseVisualStyleBackColor = true;

this.buttonClearTable.Click += new System.EventHandler(this.buttonClearTable\_Click);

//

// buttonFillTable

//

this.buttonFillTable.Location = new System.Drawing.Point(56, 83);

this.buttonFillTable.Name = "buttonFillTable";

this.buttonFillTable.Size = new System.Drawing.Size(121, 42);

this.buttonFillTable.TabIndex = 1;

this.buttonFillTable.Text = "Наповнити таблицю";

this.buttonFillTable.UseVisualStyleBackColor = true;

this.buttonFillTable.Click += new System.EventHandler(this.buttonFillTable\_Click);

//

// CountRows

//

this.CountRows.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.CountRows.Location = new System.Drawing.Point(183, 102);

this.CountRows.Name = "CountRows";

this.CountRows.Size = new System.Drawing.Size(121, 23);

this.CountRows.TabIndex = 2;

this.CountRows.Text = "500000";

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(180, 83);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(126, 13);

this.label1.TabIndex = 3;

this.label1.Text = "Ввести кількість рядків";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(307, 83);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(87, 13);

this.label2.TabIndex = 5;

this.label2.Text = "Час виконання:";

//

// timeFill

//

this.timeFill.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.timeFill.Location = new System.Drawing.Point(310, 102);

this.timeFill.Name = "timeFill";

this.timeFill.ReadOnly = true;

this.timeFill.Size = new System.Drawing.Size(121, 23);

this.timeFill.TabIndex = 4;

this.timeFill.Text = "";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(180, 131);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(87, 13);

this.label3.TabIndex = 8;

this.label3.Text = "Час виконання:";

//

// timeExecute

//

this.timeExecute.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.timeExecute.Location = new System.Drawing.Point(183, 150);

this.timeExecute.Name = "timeExecute";

this.timeExecute.ReadOnly = true;

this.timeExecute.Size = new System.Drawing.Size(121, 23);

this.timeExecute.TabIndex = 7;

this.timeExecute.Text = "";

//

// buttonExecuteQuery

//

this.buttonExecuteQuery.Location = new System.Drawing.Point(56, 131);

this.buttonExecuteQuery.Name = "buttonExecuteQuery";

this.buttonExecuteQuery.Size = new System.Drawing.Size(121, 42);

this.buttonExecuteQuery.TabIndex = 6;

this.buttonExecuteQuery.Text = "Виконати запит";

this.buttonExecuteQuery.UseVisualStyleBackColor = true;

this.buttonExecuteQuery.Click += new System.EventHandler(this.buttonExecuteQuery\_Click);

//

// buttonCreateIndex

//

this.buttonCreateIndex.Location = new System.Drawing.Point(56, 179);

this.buttonCreateIndex.Name = "buttonCreateIndex";

this.buttonCreateIndex.Size = new System.Drawing.Size(121, 42);

this.buttonCreateIndex.TabIndex = 9;

this.buttonCreateIndex.Text = "Створити індекс";

this.buttonCreateIndex.UseVisualStyleBackColor = true;

this.buttonCreateIndex.Click += new System.EventHandler(this.buttonCreateIndex\_Click);

//

// buttonDeleteIndex

//

this.buttonDeleteIndex.Location = new System.Drawing.Point(56, 227);

this.buttonDeleteIndex.Name = "buttonDeleteIndex";

this.buttonDeleteIndex.Size = new System.Drawing.Size(121, 42);

this.buttonDeleteIndex.TabIndex = 10;

this.buttonDeleteIndex.Text = "Видалити індекс";

this.buttonDeleteIndex.UseVisualStyleBackColor = true;

this.buttonDeleteIndex.Click += new System.EventHandler(this.buttonDeleteIndex\_Click);

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(180, 35);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(87, 13);

this.label4.TabIndex = 12;

this.label4.Text = "Час виконання:";

//

// timeClear

//

this.timeClear.BackColor = System.Drawing.SystemColors.Control;

this.timeClear.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.timeClear.Location = new System.Drawing.Point(183, 54);

this.timeClear.Name = "timeClear";

this.timeClear.ReadOnly = true;

this.timeClear.Size = new System.Drawing.Size(121, 23);

this.timeClear.TabIndex = 13;

this.timeClear.Text = "";

//

// timeCreate

//

this.timeCreate.BackColor = System.Drawing.SystemColors.Control;

this.timeCreate.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.timeCreate.Location = new System.Drawing.Point(183, 198);

this.timeCreate.Name = "timeCreate";

this.timeCreate.ReadOnly = true;

this.timeCreate.Size = new System.Drawing.Size(121, 23);

this.timeCreate.TabIndex = 15;

this.timeCreate.Text = "";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(180, 179);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(87, 13);

this.label5.TabIndex = 14;

this.label5.Text = "Час виконання:";

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(488, 308);

this.Controls.Add(this.timeCreate);

this.Controls.Add(this.label5);

this.Controls.Add(this.timeClear);

this.Controls.Add(this.label4);

this.Controls.Add(this.buttonDeleteIndex);

this.Controls.Add(this.buttonCreateIndex);

this.Controls.Add(this.label3);

this.Controls.Add(this.timeExecute);

this.Controls.Add(this.buttonExecuteQuery);

this.Controls.Add(this.label2);

this.Controls.Add(this.timeFill);

this.Controls.Add(this.label1);

this.Controls.Add(this.CountRows);

this.Controls.Add(this.buttonFillTable);

this.Controls.Add(this.buttonClearTable);

this.Name = "Form1";

this.Text = "Form1";

this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.Form1\_FormClosing);

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button buttonClearTable;

private System.Windows.Forms.Button buttonFillTable;

private System.Windows.Forms.RichTextBox CountRows;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.RichTextBox timeFill;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.RichTextBox timeExecute;

private System.Windows.Forms.Button buttonExecuteQuery;

private System.Windows.Forms.Button buttonCreateIndex;

private System.Windows.Forms.Button buttonDeleteIndex;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.RichTextBox timeClear;

private System.Windows.Forms.RichTextBox timeCreate;

private System.Windows.Forms.Label label5;

}

}

*“Form1.cs”*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Diagnostics;

namespace lab2

{

public partial class Form1 : Form

{

string[] FirstNames;

string[] LastNames;

string[] MiddleNames;

SqlConnection connection;

public Form1()

{

InitializeComponent();

FirstNames = new string[] { "Karen", "Dong", "Belen", "Codi", "Anneliese", "Troy", "Blaine", "Sherwood", "Arcelia", "Illa", "Shauna", "Gabrielle", "Lanora", "Neva", "Georgine", "Vivien", "Cruz", "Brandon", "Rory", "Kimbery", "Consuelo", "Deangelo", "Keiko", "Leonila", "Dannie", "Avery", "Kacey", "Hilde", "Raina", "Danette", "Tonette", "Tristan", "Gina", "Rashida", "Shawnna", "Alicia", "Loreen", "Eddie", "Kimberlie", "Roseline", "Vannessa", "Dot", "Leslie", "Clarence", "Tod", "Marcos", "Minerva", "Consuela", "Betty", "Pamula", "Carolyne", "Marlen", "Nichole", "Richie", "Tamela", "Jamee", "Suzi", "Marcellus", "Daria", "Clarisa", "George", "Trudy", "Tiny", "Suzan", "Lasonya", "Sybil", "Octavia", "Corene", "Concetta", "Titus", "Lidia", "Thanh", "Kiana", "Melodie", "Lucy", "Fletcher", "Leonie", "Trudi", "Jerrold", "Mollie", "Ashlee", "Yoshiko", "Andre", "Yoshie", "Alica", "Phylis", "Eric", "Jillian", "Artie", "Carissa", "Elizebeth", "Mable", "Jenee", "Conception", "Natasha", "Jammie", "Josette", "Wendolyn", "Norman", "Carlotta" };

LastNames = new string[] { "Hotson", "Grosvenor", "Carlyle", "De Velville", "Curzon-Howe", "Randolph", "Willhite", "Teavis", "Burrows", "Mcgettigan", "Dunlop", "Rennie", "Weatherhill", "Todhunter", "Holcroft", "Studd", "Daniell", "Toone", "Sheild", "Naumann", "Physick", "Rostron", "Nitschke", "Elkin", "Devereux", "Jenns", "Cage", "Greswolde-Williams", "Woodhouse", "Ross-Skudder", "Leybourne", "Hoks", "Cavers", "Oldfield", "Orbell", "Greer", "Herd", "Netern", "Dorothy", "Lupton", "Donnely", "Lester", "Fenner", "Cron", "Hewson", "Redhead", "Blanks", "Neelin", "Murray", "Theakston", "Festing", "Goodeve", "Nethercote", "Wilmoth", "Muirhead", "Trippett", "Huddlestone", "Salman", "Elliott", "Honeyman", "Macallister", "Sim", "Thomlinson", "Pickles", "Kinnish", "Marfleet", "Bottoms", "Vanstan", "De Aldburgh", "Gilbraith", "Jensen", "Kelland", "Tedford", "Saul", "Pearl", "Ferrant", "Livsey", "Rule", "Whittington", "Corton", "Nicklenson", "Edmunds", "Macfie", "Flinders", "Dorynne", "Cresmer", "Dawtry", "Plant", "Wriggle", "Laroche", "Chisnall", "Lowrie", "Gall", "Bell", "Druistan", "Sowerby Battye", "Milbourne", "Glencross", "Robey", "Stott" };

MiddleNames = new string[] { "Alexeyevich", "Nikitevich", "Denisovich", "Andreevich", "Onisimovich", "Sidorovich", "Leonidovich", "Ernstovich", "Artemievich", "Erofeevich", "Filimonovich", "Ipatovich", "Epifanovich", "Serafimovich", "Pakhomovich", "Glebovich", "Filimonovich", "Onisimovich", "Andreevich", "Svyatoslavovich", "Dmitrievich", "Eremeevich", "Arsenievich", "Gerasimovich", "Davydovich", "Artemievich", "Feliksovich", "Kirillovich", "Vyacheslavovich", "Klimentovych", "Grigoryevich", "Socratovich", "Nikanorovich", "Modestovic", "Jakowowicz", "Nikitevich", "Ostapovich", "Gordeyevich", "Nikiforovich", "Daniilovich", "Izmailovich", "Semenovich", "Makarovich", "Valeryanovich", "Prohorovich", "Adamovich", "Samuilovich", "Grigoryevich", "Feliksovich", "Valerievich", "Vladimirovich", "Glebovich", "Yelizarovich", "Nestorovich", "Vladislavovich", "Danilevich", "Tikhonovich", "Levovich", "Leonidovich", "Valerievich", "Ippolitovich", "Kondratievich", "Epifanovich", "Onisimovich", "Kondratievich", "Andronovych", "Savelievich", "Prokilovich", "Sidorovich", "Pankratievich", "Karlovich", "Ipatievich", "Venediktovych", "Sigismundovich", "Gavrilevich", "Feliksovich", "Afanasyevich", "Klimentovych", "Adrianovich", "Czeslawowicz", "Sidorovich", "Mikhailovich", "Vadimovich", "Nikitevich", "Nikonovic", "Andronikovich", "Agapovich", "Iraklievich", "Igorevich", "Tarasovich", "Klimentovych", "Andronikovich", "Moiseevich", "Semenovich", "Olegovych", "Nestorovich", "Kazimirovich", "Adrianovich", "Zinovievich", "Socratovich" };

connection = new SqlConnection(@"Data Source = VALENTINE\SQLEXPRESS; Initial Catalog = Kpi2; Integrated Security = True");

connection.Open();

string queryString =

"IF EXISTS(SELECT \* FROM SYSOBJECTS WHERE NAME = 'Complaints' AND xtype = 'U')" +

"DROP TABLE [Complaints];" +

"IF EXISTS(SELECT \* FROM SYSOBJECTS WHERE NAME = 'Clients' AND xtype = 'U')" +

"DROP TABLE [Clients];" +

"CREATE TABLE[Clients](" +

"[ClientID] INT IDENTITY(1,1) NOT NULL," +

"[FirstName] NVARCHAR(255) NOT NULL," +

"[LastName] NVARCHAR(255) NOT NULL," +

"[MiddleName] NVARCHAR(255)NOT NULL," +

"CONSTRAINT PK\_ClientID PRIMARY KEY CLUSTERED(ClientID))" +

"CREATE TABLE [Complaints] (" +

"[ComplaintID] INT IDENTITY(1,1) NOT NULL," +

"[Price] NVARCHAR(255) NOT NULL," +

"[ClientID] INT NOT NULL," +

"CONSTRAINT FK\_ClientID FOREIGN KEY(ClientID) REFERENCES[Clients]([ClientID])," +

"CONSTRAINT PK\_ComplaintID PRIMARY KEY CLUSTERED(ComplaintID))";

SqlCommand command = new SqlCommand(queryString, connection);

command.ExecuteNonQuery();

connection.Close();

}

private void buttonClearTable\_Click(object sender, EventArgs e)

{

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

connection.Open();

string queryString =

"TRUNCATE TABLE [Complaints] " +

"ALTER TABLE [Complaints] " +

"DROP CONSTRAINT FK\_ClientID " +

"TRUNCATE TABLE [Clients]" +

"ALTER TABLE [Complaints] " +

"ADD CONSTRAINT FK\_ClientID FOREIGN KEY(ClientID) REFERENCES[Clients]([ClientID])";

SqlCommand command = new SqlCommand(queryString, connection);

command.ExecuteNonQuery();

connection.Close();

stopWatch.Stop();

timeClear.Text = (stopWatch.ElapsedMilliseconds / 1000.0).ToString() + "s";

}

private void buttonFillTable\_Click(object sender, EventArgs e)

{

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

connection.Open();

int rowCount = Convert.ToInt32(CountRows.Text);

Random random = new Random();

DataTable table1 = new DataTable();

table1.Columns.Add("ClientID", typeof(int));

table1.Columns.Add("FirstName", typeof(string));

table1.Columns.Add("LastName", typeof(string));

table1.Columns.Add("MiddleName", typeof(string));

for (int i = 0; i < rowCount; i++)

table1.Rows.Add(new object[] {

i + 1,

FirstNames[random.Next(0, 99)],

LastNames[random.Next(0, 99)],

MiddleNames[random.Next(0, 99)] });

SqlBulkCopy bulkCopy1 = new SqlBulkCopy(connection);

bulkCopy1.DestinationTableName = "[Clients]";

bulkCopy1.WriteToServer(table1);

connection.Close();

connection.Open();

DataTable table2 = new DataTable();

table2.Columns.Add("ComplaintID", typeof(int));

table2.Columns.Add("Price", typeof(string));

table2.Columns.Add("ClientID", typeof(int));

for (int i = 0; i < rowCount; i++)

table2.Rows.Add(new object[] {

i + 1,

((10 \* random.Next(0, 1000)).ToString() + "$"),

random.Next(1, rowCount) });

SqlBulkCopy bulkCopy2 = new SqlBulkCopy(connection);

bulkCopy2.DestinationTableName = "[Complaints]";

bulkCopy2.WriteToServer(table2);

connection.Close();

stopWatch.Stop();

timeFill.Text = (stopWatch.ElapsedMilliseconds / 1000.0).ToString() + "s";

}

private void buttonExecuteQuery\_Click(object sender, EventArgs e)

{

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

connection.Open();

string queryString =

"SELECT [FirstName]," +

"[LastName]," +

"[MiddleName]" +

"[Price]" +

"FROM [Complaints] AS [CO]" +

"JOIN [Clients] AS [CL]" +

" ON CO.ClientID = CL.ClientID " +

"WHERE [Price] = '1000$'";

SqlCommand command = new SqlCommand(queryString, connection);

command.ExecuteNonQuery();

connection.Close();

stopWatch.Stop();

timeExecute.Text = (stopWatch.ElapsedMilliseconds / 1000.0).ToString() + "s";

}

private void buttonCreateIndex\_Click(object sender, EventArgs e)

{

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

connection.Open();

string queryString =

"IF NOT EXISTS(SELECT \* FROM sysindexes WHERE name = 'Index1')" +

"CREATE NONCLUSTERED INDEX Index1 ON [Complaints]([Price])";

SqlCommand command = new SqlCommand(queryString, connection);

command.ExecuteNonQuery();

connection.Close();

stopWatch.Stop();

timeCreate.Text = (stopWatch.ElapsedMilliseconds / 1000.0).ToString() + "s";

}

private void buttonDeleteIndex\_Click(object sender, EventArgs e)

{

connection.Open();

string queryString =

"IF EXISTS(SELECT \* FROM sysindexes WHERE name = 'Index1')" +

"DROP INDEX [Index1] ON [dbo].[Complaints]";

SqlCommand command = new SqlCommand(queryString, connection);

command.ExecuteNonQuery();

connection.Close();

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

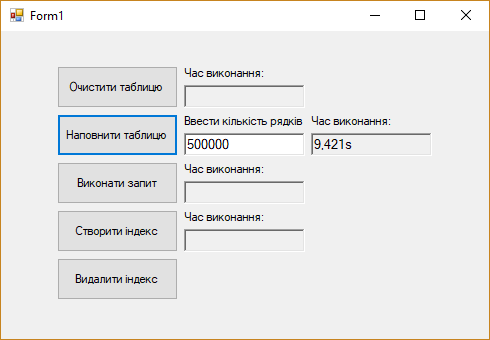
connection.Dispose();

}

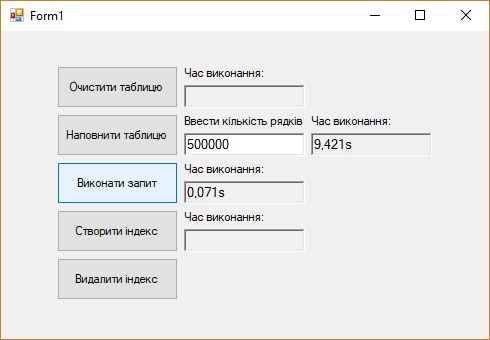
}

}

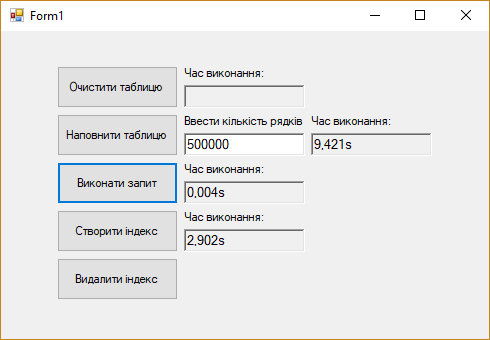
*Результат виконання програми після натискання кнопки «Наповнити таблицю»*:



*Результат виконання програми після натискання кнопки «Виконати запит»*:



*Результат виконання програми після натискання кнопок «Створити індекс» та «Виконати запит»*:



1. Індекси:
   1. Б-дерева

Алгоритми, що працюють з Б-деревами, зберігають в оперативній пам'яті лише невелику частину всієї інформації (фіксоване число секторів).

Диск розглядається як велика ділянка пам'яті, робота з яким відбувається наступним чином: перед тим як працювати з об'єктом x, виконується спеціальна операція Disk — Read(x) (читання з диска). Після внесення змін в об'єкт x виконується операція Disk — Write(x) (запис на диск).

Час роботи програми в основному визначається кількістю цих операцій, так що має сенс читати / записувати як можна більше інформації за один раз і зробити так, щоб вузол Б-дерева заповнював повністю один сектор диска. Таким чином, ступінь розгалуження (число дітей вузла) визначається розміром сектора.

Типова ступінь розгалуження Б-дерев знаходиться між 50 і 2000 в залежності від розміру елемента. Збільшення ступеня розгалуження різко скорочує висоту дерева, і тим самим число звернень до диску, при пошуку. Наприклад, Б-дерево ступеня 1001 і висоти 2 може зберігати понад мільярд ключів. Враховуючи, що корінь можна постійно зберігати в оперативній пам'яті, достатньо двох звернень до диска при пошуку потрібного ключа.

Вважаємо, що прикладна інформація, пов'язана з ключем, зберігається в тому ж вузлі дерева. На практиці це не завжди зручно, і в реальному алгоритмі вузол може містити лише посилання на сектор, де вона зберігається.

**Визначення Б-дерева**

**Б-деревом** називають кореневе дерево, влаштоване наступним чином. Кожен вузол x містить наступні поля:

* n[x] — кількість ключів, що зберігаються у вузлі x;
* key1[x], key2[x], … ,keyn(x)[x] — самі ключі в не спадаючому порядку;
* leaf[x] — булеве значення, істинне, коли вузол x є листом.

Якщо x — внутрішній вузол, то він містить покажчики c1[x], c2[x] cn(x)+1[x], на його дітей в кількості n[x]+1.

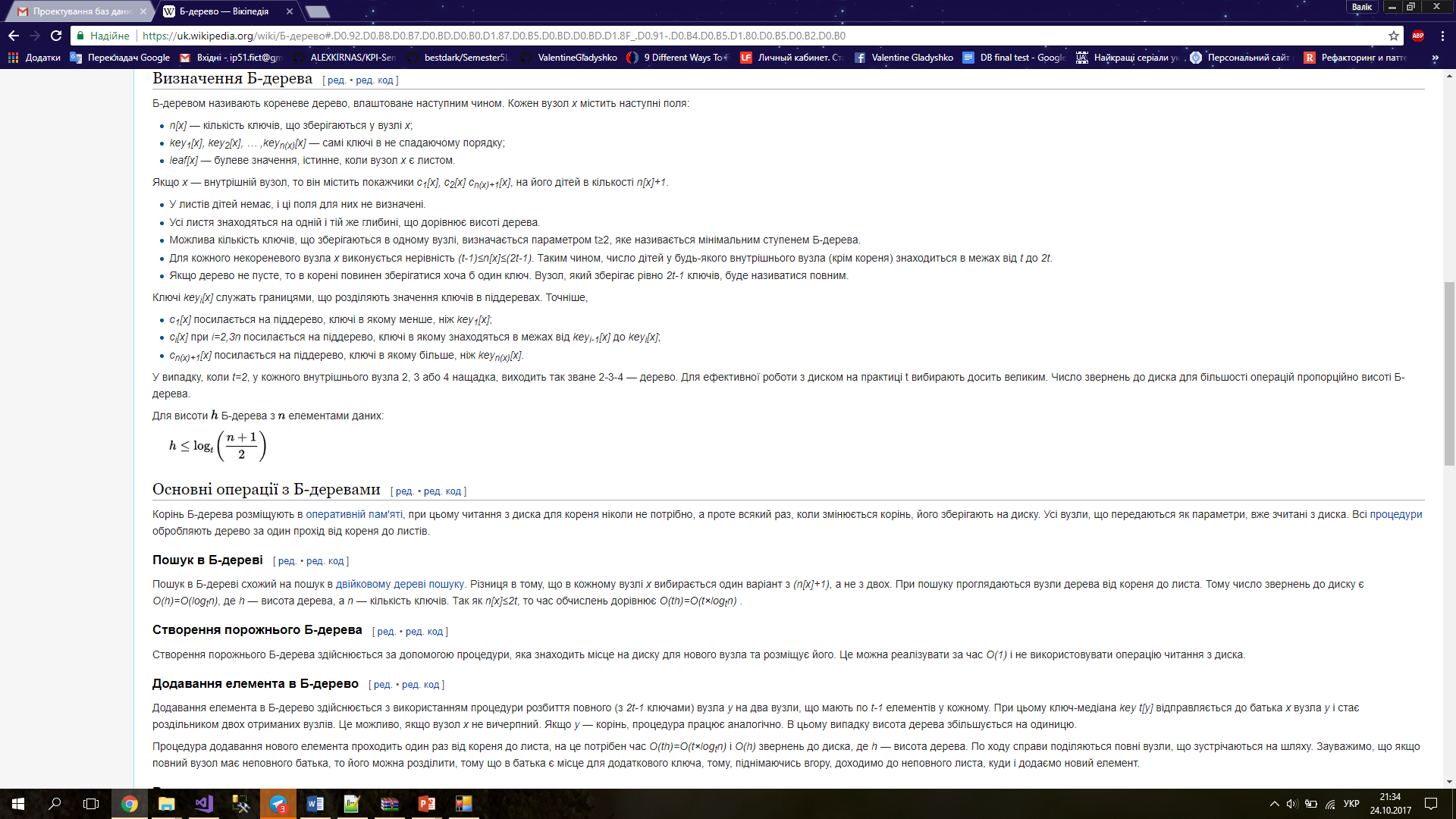
* У листів дітей немає, і ці поля для них не визначені.
* Усі листя знаходяться на одній і тій же глибині, що дорівнює висоті дерева.
* Можлива кількість ключів, що зберігаються в одному вузлі, визначається параметром t≥2, яке називається мінімальним ступенем Б-дерева.
* Для кожного некореневого вузла x виконується нерівність (t-1)≤n[x]≤(2t-1). Таким чином, число дітей у будь-якого внутрішнього вузла (крім кореня) знаходиться в межах від t до 2t.
* Якщо дерево не пусте, то в корені повинен зберігатися хоча б один ключ. Вузол, який зберігає рівно 2t-1 ключів, буде називатися повним.

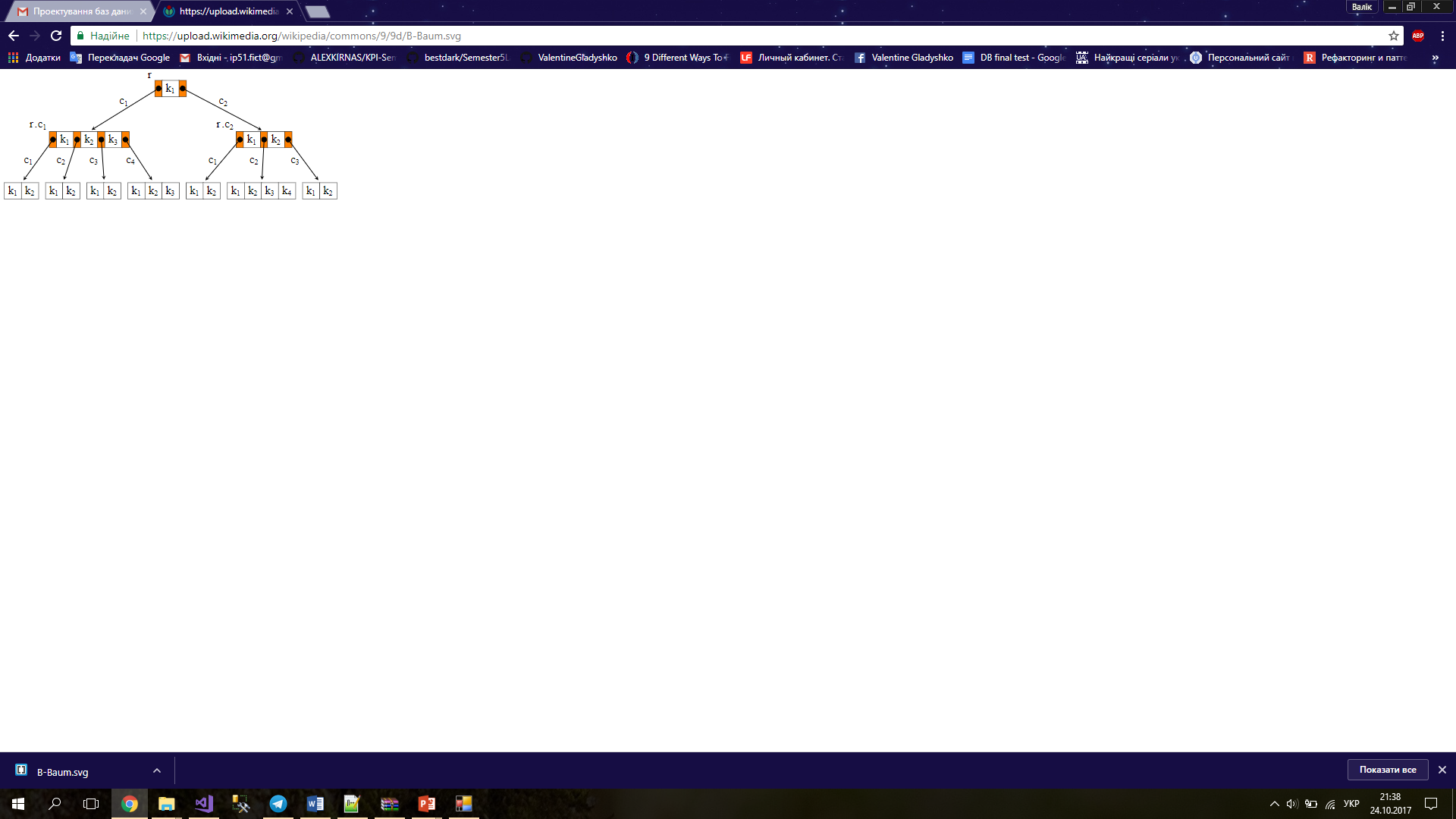
Ключі keyi[x] служать границями, що розділяють значення ключів в піддеревах. Точніше,

* c1[x] посилається на піддерево, ключі в якому менше, ніж key1[x];
* ci[x] при i=2,3n посилається на піддерево, ключі в якому знаходяться в межах від keyi-1[x] до keyi[x];
* cn(x)+1[x] посилається на піддерево, ключі в якому більше, ніж keyn(x)[x].

У випадку, коли t=2, у кожного внутрішнього вузла 2, 3 або 4 нащадка, виходить так зване 2-3-4 — дерево. Для ефективної роботи з диском на практиці t вибирають досить великим. Число звернень до диска для більшості операцій пропорційно висоті Б-дерева.

Для висоти h Б-дерева з n елементами даних:





**Основні операції з Б-деревами**

Корінь Б-дерева розміщують в оперативній пам'яті, при цьому читання з диска для кореня ніколи не потрібно, а проте всякий раз, коли змінюється корінь, його зберігають на диску. Усі вузли, що передаються як параметри, вже зчитані з диска. Всі процедури обробляють дерево за один прохід від кореня до листів.

* Пошук в Б-дереві

Пошук в Б-дереві схожий на пошук в двійковому дереві пошуку. Різниця в тому, що в кожному вузлі x вибирається один варіант з (n[x]+1), а не з двох. При пошуку проглядаються вузли дерева від кореня до листа. Тому число звернень до диску є O(h)=O(logtn), де h — висота дерева, а n — кількість ключів. Так як n[x]≤2t, то час обчислень дорівнює O(th)=O(t×logtn) .

* Створення порожнього Б-дерева

Створення порожнього Б-дерева здійснюється за допомогою процедури, яка знаходить місце на диску для нового вузла та розміщує його. Це можна реалізувати за час O(1) і не використовувати операцію читання з диска.

* Додавання елемента в Б-дерево

Додавання елемента в Б-дерево здійснюється з використанням процедури розбиття повного (з 2t-1 ключами) вузла y на два вузли, що мають по t-1 елементів у кожному. При цьому ключ-медіана key t[y] відправляється до батька x вузла y і стає роздільником двох отриманих вузлів. Це можливо, якщо вузол x не вичерпний. Якщо y — корінь, процедура працює аналогічно. В цьому випадку висота дерева збільшується на одиницю.

Процедура додавання нового елемента проходить один раз від кореня до листа, на це потрібен час O(th)=O(t×logtn) і O(h) звернень до диска, де h — висота дерева. По ходу справи поділяються повні вузли, що зустрічаються на шляху. Зауважимо, що якщо повний вузол має неповного батька, то його можна розділити, тому що в батька є місце для додаткового ключа, тому, піднімаючись вгору, доходимо до неповного листа, куди і додаємо новий елемент.

* Видалення елемента

Видалення елемента з B-дерева відбувається аналогічно додаванню, хоча трохи складніше. Видалення ключа з В-дерева, хоча і аналогічно вставці, являє собою складнішу задачу. Це пов'язано з тим, що ключ може бути видалений з будь-якого вузла, а не тільки з листа, а видалення з внутрішнього вузла вимагає певної перебудови дочірніх вузлів. Як і у випадку вставки, ми повинні забезпечити, щоб при виконанні операції видалення не були порушені властивості В-дерева. Аналогічно тому, як ми мали можливість переконатися, що вузли не надто сильно заповнені для вставки нового ключа, нам належить переконатися, що вузол не стає занадто мало заповнений в процесі видалення ключа (за винятком кореневого вузла, який може мати менше t — 1 ключів, хоча і не може мати більше 2t — 1 ключів).

Отже, нехай процедура B\_TREE\_DELETE повинна видалити ключ k з піддереві, коренем якого є вузол x. Ця процедура розроблена таким чином, що при її рекурсивному виклику для вузла х гарантована наявність в цьому вузлі принаймні t ключів. Ця умова вимагає наявності у вузлі більшої кількості ключів, ніж мінімальна в звичайному В-дереві, так що іноді ключ може бути переміщений в дочірній вузол перед тим, як рекурсія звернеться до цього дочірньому вузлу. Таке посилення властивості В-дерева (наявність «запасного» ключа) дає нам можливість виконати видалення ключа за один спадний прохід по дереву (з єдиним винятком, який буде пояснено пізніше). Слід також врахувати, що якщо корінь дерева х стає внутрішнім вузлом, що не містить жодного ключа (така ситуація може виникнути в розглянутих нижче випадках 2в і 36), то вузол х віддаляється, а його єдиний дочірній вузол С1[х] стає новим коренем дерева (при цьому зменшується висота В-дерева і зберігається його властивість, що вимагає, щоб кореневий вузол непорожнього дерева містив як мінімум один ключ).

Замість того щоб представити вам повний псевдокод процедури видалення вузла з В-дерева, ми просто накидаємо послідовність виконуваних дій.

1. Якщо вузол k знаходиться у вузлі x і x є листом — видаляємо k з х.

2. Якщо вузол k знаходиться у вузлі х і х є внутрішнім вузлом, виконуємо наступні дії:

а) Якщо дочірній по відношенню до х вузол у, попередній ключу k у вузлі x, містить не менше t ключів, то знаходимо до k' попередника k в піддереві, коренем якого є у. Рекурсивно видаляємо k' і замінюємо k в х ключем k'. (Пошук ключа k' і видалення його можна виконати в процесі одного спадного проходу.)

б) Ситуація, симетрична ситуації а: якщо дочірній по відношенню до х вузол z, наступний за ключем k в вузлі x, містить не менше t ключів, то знаходимо k' — наступний за k ключ в піддереві, коренем якого є z. Рекурсивно видаляємо k' і замінюємо k в х ключем k'. (Пошук ключа k' і видалення його можна виконати в процесі одного спадного проходу.)

в) У противному випадку, якщо і y, і z містять по t — 1 ключів, вносимо k і всі ключі z в у (при цьому з х видаляється k і покажчик на z, а вузол у після цього містить 2t — 1 ключів), а потім звільняємо z і рекурсивно видаляємо k з у.

3. Якщо ключ k відсутній у внутрішньому вузлі х, знаходимо корінь cі[х] піддерева, яке повинно містити k (якщо такий ключ є в даному В-дереві). Якщо cі[х] містить тільки t — 1 ключів, виконуємо крок 3а або 3б для того, щоб гарантувати, що далі ми переходимо в вузол, що містить як мінімум t ключів. Потім ми рекурсивно видаляємо k з піддерева з коренем cі[х].

а) Якщо cі[х] містить тільки t — 1 ключів, але при цьому один з її безпосередніх сусідів (під яким ми розуміємо дочірній по відношенню до х вузол, відокремлений від розглянутого рівно одним ключем-роздільником) містить як мінімум t ключів, передамо в cі[х] ключ-роздільник між даними вузлом і його безпосереднім сусідом з x, на його місце помістимо крайній ключ із сусіднього вузла і перенесемо відповідний покажчик із сусіднього вузла в cі[х].

б) Якщо і cі[х] і обидва його безпосередніх сусіда містять по t — 1 ключів, об'єднаємо cі[х] з одним з його сусідів (при цьому колишній ключ-роздільник з x стане медіаною нового вузла).

* 1. Для яких запитів який тип індексу використовує вибрана вами СУБД

Обрана мною СУБД для всіх операцій пошуку, порівняння, при виконанні операції WHERE, BETWEEN використовую некластерний індекс який базується на B-деревах (за умови що по даному полю побудований індекс).

1. **ВИСНОВОК**

Отже, я навчився використовувати бази даних в програмних додатках. Також я ознайомився з основними видами індексів: кластерний та некластерний та їхньою структурою: Б-дерево. Я використав ці знання щоб прискорити виконання запитів за допомогою некластерного індексу.