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 6

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Cт.вик. Клименко О. М. |  | студент 3-го курсу  гр. ІП-51 ФІОТ  Булатов Дмитро Єгорович |

Київ – 2017

**ЗМІСТ:**

[1 Завдання 3](#_Toc495230724)

[2 Виконання Завдання 4](#_Toc495230725)

# 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**: Навчання з охорони праці

# Виконання Завдання

1. Завдання по написанню запитів

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Reflection.Emit;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace OSLab2

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

var connection = new SqlConnection("Data Source=(local);Integrated Security=SSPI;");

var StartTime = DateTime.Now;

connection.Open();

var command = new SqlCommand("USE LR2; DELETE FROM Rent; DELETE FROM Client", connection);

command.ExecuteNonQuery();

TimeSpan diff = DateTime.Now - StartTime;

TimeLabel.Content = $"completed in {diff.TotalSeconds} seconds";

var getCount = new SqlCommand("USE LR2; SELECT COUNT(\*) FROM Client", connection);

int CountRows = Convert.ToInt32(getCount.ExecuteScalar());

CountLabel.Content = CountRows.ToString() + " rows";

connection.Close();

InfoBox.Items.Clear();

}

public static readonly string[] firstNames = { "Dmytro", "Ivan", "Ilya", "Anastasia", "Sasha", "Svetlana", "Alina", "Julia", "Anna", "Stanislav"};

public static readonly string[] secondNames = { "Bulatov", "Volkov", "Voitenko", "Starchenko", "Yadelskiy", "Reutska", "Melnikova", "Gamova", "Khuda", "Snitsarenko" };

public static readonly string[] middleNames = { "Dmytrovich", "Ivanov", "Illich", "Stanislavovich", "Aleksandrovich", "Konstantinovich", "Vadimovich", "Egorovich", "Valeriovich", "Vadimovich" };

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

var connection = new SqlConnection("Data Source=(local);Integrated Security=SSPI;");

connection.Open();

var StartTime = DateTime.Now;

int C1 = Convert.ToInt32(NumInput.Text);

int Count = C1 < 1000 ? C1 : 1000;

var getLastID = new SqlCommand("USE LR2; SELECT TOP(1) ID FROM Client ORDER BY ID DESC", connection);

int ID = Convert.ToInt32(getLastID.ExecuteScalar()) + 1;

while (C1 > 0)

{

C1 -= Count;

string sqlQuery = "USE LR2; INSERT INTO Client(ID,FirstName,LastName,MiddleName) VALUES ";

string sqlQuery2 = "USE LR2; INSERT INTO Rent(ID,ClientID) VALUES ";

var r = new Random();

for (; Count > 0; Count--, ID++)

{

sqlQuery +=

$"({ID},'{firstNames[r.Next() % 10]}','{secondNames[r.Next() % 10]}','{middleNames[r.Next() % 10]}')";

if (Count != 1) sqlQuery += ",";

sqlQuery2 +=

$"({ID},{ID})";

if (Count != 1) sqlQuery2 += ",";

}

var command = new SqlCommand(sqlQuery, connection);

command.ExecuteNonQuery();

var command2 = new SqlCommand(sqlQuery2, connection);

command2.ExecuteNonQuery();

Count = C1 < 1000 ? C1 : 1000;

}

var select = new SqlCommand("USE LR2; SELECT TOP(1000) ID,FirstName,LastName,MiddleName FROM Client ORDER BY Client.ID DESC", connection);

var reader = select.ExecuteReader();

InfoBox.Items.Clear();

if (reader.HasRows)

{

while (reader.Read())

{

InfoBox.Items.Add(reader.GetString(1) + " " + reader.GetString(2) + " " + reader.GetString(3));

}

}

reader.Close();

TimeSpan diff = DateTime.Now - StartTime;

TimeLabel.Content = $"completed in {diff.TotalSeconds} seconds";

var getCount = new SqlCommand("USE LR2; SELECT COUNT(\*) FROM Client", connection);

int CountRows = Convert.ToInt32(getCount.ExecuteScalar());

CountLabel.Content = CountRows.ToString() + " rows";

connection.Close();

}

private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void ListView\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

private void Grid\_Initialized(object sender, EventArgs e)

{

var connection = new SqlConnection("Data Source=(local);Integrated Security=SSPI;");

connection.Open();

var select = new SqlCommand("USE LR2; SELECT TOP(1000) ID,FirstName,LastName,MiddleName FROM Client ORDER BY ID DESC", connection);

var reader = select.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

InfoBox.Items.Add(reader.GetString(1) + " " + reader.GetString(2) + " " + reader.GetString(3));

}

}

reader.Close();

var getCount = new SqlCommand("USE LR2; SELECT COUNT(\*) FROM Client", connection);

int CountRows = Convert.ToInt32(getCount.ExecuteScalar());

CountLabel.Content = CountRows.ToString() + " rows";

connection.Close();

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

var connection = new SqlConnection("Data Source=(local);Integrated Security=SSPI;");

connection.Open();

var StartTime = DateTime.Now;

var select = new SqlCommand($"USE LR2; SELECT Client.ID,FirstName,LastName,MiddleName,StartDate FROM Client WITH (INDEX(ID\_Index)) JOIN Rent ON Rent.ClientID = Client.ID ORDER BY Client.ID DESC", connection);

var reader = select.ExecuteReader();

TimeSpan diff = DateTime.Now - StartTime;

TimeLabel.Content = $"completed in {diff.TotalSeconds} seconds";

int i = 0;

InfoBox.Items.Clear();

if (reader.HasRows)

{

while (reader.Read() && i < 1000)

{

InfoBox.Items.Add(reader.GetString(1) + " " + reader.GetString(2) + " " + reader.GetString(3) + " " + ((System.DateTime)reader[4]).ToShortTimeString());

i++;

}

}

reader.Close();

var getCount = new SqlCommand("USE LR2; SELECT COUNT(\*) FROM Client", connection);

int CountRows = Convert.ToInt32(getCount.ExecuteScalar());

CountLabel.Content = CountRows.ToString() + " rows";

connection.Close();

}

private void Button\_Click\_4(object sender, RoutedEventArgs e)

{

var connection = new SqlConnection("Data Source=(local);Integrated Security=SSPI;");

connection.Open();

var StartTime = DateTime.Now;

var getLastID = new SqlCommand("USE LR2; SELECT TOP(1) ID FROM Client ORDER BY ID DESC", connection);

int ID = Convert.ToInt32(getLastID.ExecuteScalar()) + 1;

var r = new Random();

var select = new SqlCommand($"USE LR2; SELECT Client.ID,FirstName,LastName,MiddleName FROM Client WITH (INDEX(ID\_Index)) WHERE ID = {r.Next() % ID + 1} ORDER BY Client.ID DESC", connection);

var reader = select.ExecuteReader();

TimeSpan diff = DateTime.Now - StartTime;

TimeLabel.Content = $"completed in {diff.TotalSeconds} seconds";

int i = 0;

InfoBox.Items.Clear();

if (reader.HasRows)

{

while (reader.Read() && i < 1000)

{

InfoBox.Items.Add(reader.GetString(1) + " " + reader.GetString(2) + " " + reader.GetString(3));

i++;

}

}

reader.Close();

var getCount = new SqlCommand("USE LR2; SELECT COUNT(\*) FROM Client", connection);

int CountRows = Convert.ToInt32(getCount.ExecuteScalar());

CountLabel.Content = CountRows.ToString() + " rows";

connection.Close();

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

var connection = new SqlConnection("Data Source=(local);Integrated Security=SSPI;");

connection.Open();

var StartTime = DateTime.Now;

var select = new SqlCommand($"USE LR2; SELECT Client.ID,FirstName,LastName,MiddleName,StartDate FROM Client WITH (INDEX(0)) JOIN Rent ON Rent.ClientID = Client.ID ORDER BY Client.ID DESC", connection);

var reader = select.ExecuteReader();

TimeSpan diff = DateTime.Now - StartTime;

TimeLabel.Content = $"completed in {diff.TotalSeconds} seconds";

int i = 0;

InfoBox.Items.Clear();

if (reader.HasRows)

{

while (reader.Read() && i < 1000)

{

InfoBox.Items.Add(reader.GetString(1) + " " + reader.GetString(2) + " " + reader.GetString(3) + " " + ((System.DateTime)reader[4]).ToShortTimeString());

i++;

}

}

reader.Close();

var getCount = new SqlCommand("USE LR2; SELECT COUNT(\*) FROM Client", connection);

int CountRows = Convert.ToInt32(getCount.ExecuteScalar());

CountLabel.Content = CountRows.ToString() + " rows";

connection.Close();

}

private void Button\_Click\_5(object sender, RoutedEventArgs e)

{

var connection = new SqlConnection("Data Source=(local);Integrated Security=SSPI;");

connection.Open();

var StartTime = DateTime.Now;

var getLastID = new SqlCommand("USE LR2; SELECT TOP(1) ID FROM Client ORDER BY ID DESC", connection);

int ID = Convert.ToInt32(getLastID.ExecuteScalar()) + 1;

var r = new Random();

var select = new SqlCommand($"USE LR2; SELECT Client.ID,FirstName,LastName,MiddleName FROM Client WITH (INDEX(0)) WHERE ID = {r.Next() % ID + 1} ORDER BY Client.ID DESC", connection);

var reader = select.ExecuteReader();

TimeSpan diff = DateTime.Now - StartTime;

TimeLabel.Content = $"completed in {diff.TotalSeconds} seconds";

int i = 0;

InfoBox.Items.Clear();

if (reader.HasRows)

{

while (reader.Read() && i < 1000)

{

InfoBox.Items.Add(reader.GetString(1) + " " + reader.GetString(2) + " " + reader.GetString(3));

i++;

}

}

reader.Close();

var getCount = new SqlCommand("USE LR2; SELECT COUNT(\*) FROM Client", connection);

int CountRows = Convert.ToInt32(getCount.ExecuteScalar());

CountLabel.Content = CountRows.ToString() + " rows";

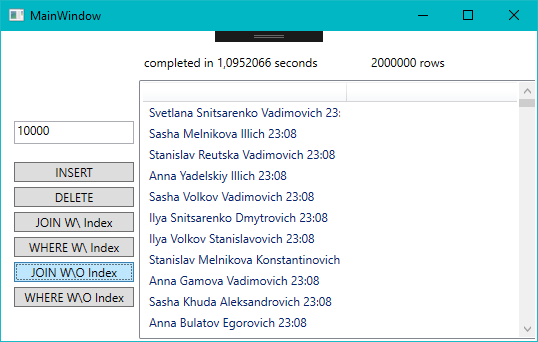
connection.Close();

}

}

}

*Результат виконання програми*:



1. Індекси:
   1. Описати які типи індексів використовує вибрана вами СУБД і як кожен тип реалізований алгоритмічно (наприклад, B-tree індекс та принцип його роботи).

MS SQL завжди використовує бі-дерева

**B-дерево** (англ. *B-tree*) — сильноветвящееся сбалансированное дерево поиска, позволяющее проводить поиск, добавление и удаление элементов за O(\log n). B-дерево с n узлами имеет высоту O(\log n). Количество детей узлов может быть от нескольких до тысяч (обычно степень ветвления B-дерева определяется характеристиками устройства (дисков), на котором производится работа с деревом). В-деревья также могут использоваться для реализации многих операций над динамическими множествами за время O(\log n)

B-дерево является идеально сбалансированным, то есть глубина всех его листьев одинакова. B-дерево имеет следующие свойства (t — параметр дерева, называемый *минимальной степенью* B-дерева, не меньший 2.):

* Каждый узел, кроме корня, содержит не менее t - 1 ключей, и каждый внутренний узел имеет по меньшей мере t дочерних узлов. Если дерево не является пустым, корень должен содержать как минимум один ключ.
* Каждый узел, кроме корня, содержит не более 2t - 1 ключей и не более чем 2t сыновей во внутренних узлах
* Корень содержит от 1 до 2t - 1 ключей, если дерево не пусто и от 2 до 2t детей при высоте большей 0.
* Каждый узел дерева, кроме листьев, содержащий ключи k_1, ..., k_n, имеет n + 1 сына. i-й сын содержит ключи из отрезка [k_{i - 1}; k_i],\:  k_0 = -\infty,\: k_{n + 1} = \infty.
* Ключи в каждом узле упорядочены по неубыванию.
* Все листья находятся на одном уровне.

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

MS SQL для усіх операцій використовує некластерний індекс який базується на B-деревах (якщо по даному полю побудований індекс).