министерство НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
Российской федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Тюменский государственный университет»

институт математики и компьютерных наук

Кафедра программного обеспечения

Курсовая работа по направлению

«Математическое обеспечение и администрирование

информационных систем»

на тему «Разработка ПО для резервного копирования и восстановления баз данных MS SQL Server»

Выполнил: студент 3 курса

22 МОиАИС 184-2 группы

Поляков И.А

Научный руководитель:

доцент кафедры ПО

Ялдыгин В.Б

Тюмень 2021

**Содержание**

[Введение 3](file:///C:\Users\augus\Downloads\DataBase2_8901237%20(1)\DataBase\Курсовая.docx#_Toc82616065)

[Глава 1. Основные принципы резервного копирования и восстановления баз данных SQL Server 4](file:///C:\Users\augus\Downloads\DataBase2_8901237%20(1)\DataBase\Курсовая.docx#_Toc82616066)

[Глава 2. Создание программного обеспечения для резервного копирование и восстановления базы данных. 8](file:///C:\Users\augus\Downloads\DataBase2_8901237%20(1)\DataBase\Курсовая.docx#_Toc82616067)

[Глава 3. Графическое описание программного обеспечения 18](file:///C:\Users\augus\Downloads\DataBase2_8901237%20(1)\DataBase\Курсовая.docx#_Toc82616068)

[Глава 4. Тестирование работы программного обеспечения 19](file:///C:\Users\augus\Downloads\DataBase2_8901237%20(1)\DataBase\Курсовая.docx#_Toc82616069)

[Заключение 23](file:///C:\Users\augus\Downloads\DataBase2_8901237%20(1)\DataBase\Курсовая.docx#_Toc82616070)

[Список используемых источников 24](file:///C:\Users\augus\Downloads\DataBase2_8901237%20(1)\DataBase\Курсовая.docx#_Toc82616071)

Введение

В настоящее время безопасное хранение данных представляет одну из основных проблем информационной безопасности. Согласно докладу о рисках Всемирного экономического форума 2021 года, утеря и утечка данных являются четвертым и пятым глобальными рисками, с которыми сталкивается каждая организация. По своей значимости эти риски приравниваются к экологическим проблемам. В официальном ежегодном отчете за 2019 год, опубликованном Cybersecurity Ventures, сообщается, что атаки хакеров во всём мире происходят каждые 11 секунд. Специалисты компании InfoWatch в конце года рассказали Известиям, что за прошлый год в сеть утекло более 14 млрд. конфиденциальных записей. Рост числа утечек во всём мире по сравнению с 2018 годом увеличился на 10%, в России – более чем на 40%.

Их этого можно сделать вывод, что хранение данных, должно обеспечивать их целостность. Большинство данных хранятся в базах, так как это является удобным способом взаимодействия с ними. Одной из основных технологий баз данных является Sql Server, который является системой управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка. Для сохранения целостности баз данных используют такие технологии как их резервное копирование и восстановление.

Глава 1. Основные принципы резервного копирования и восстановления баз данных SQL Server

Резервное копирование базы данных - это процесс загрузки данных (из базы данных, журнала транзакций или файла) на устройства резервного копирования, которые создаются и обслуживаются системой. Устройство резервного копирования может быть дисковым файлом или магнитной лентой. Компонент Database Engine обеспечивает статическое и динамическое резервное копирование. Статическое резервное копирование означает, что во время процесса резервного копирования только один активный сеанс, поддерживаемый системой, является сеансом, создающим резервную копию. Другими словами, во время копирования никакие пользовательские процессы не допускаются. Динамическое резервное копирование означает, что копирование базы данных может выполняться без остановки сервера базы данных, удаления пользователей и даже без закрытия файлов. (Пользователи даже не узнают, что идет процесс резервного копирования.)

Компонент Database Engine предоставляет четыре различных метода резервного копирования:

1. Полное копирование базы данных;
2. Дифференцированное резервное копирование;
3. Резервное копирование протокола транзакции;
4. Резервное копирование файла или файловой группы.

Полная копия базы данных фиксирует состояние базы данных, каким оно было при запуске копии. Во время полного копирования базы данных система копирует данные, а также схему всех таблиц в базе данных и соответствующие файловые структуры. Если полная копия базы данных выполняется динамически, система базы данных записывает все действия, происходящие во время процесса резервного копирования. Поэтому даже все неподтвержденные транзакции в журнале транзакций будут записаны на устройство резервного копирования.

Все типы резервных копий можно выполнять с помощью двух операторов Transact-SQL:

1. backup database;
2. backup log.

Компонент Database Engine позволяет копировать базы данных, журналы транзакций и файлы на устройства резервного копирования, такие как:

1. Диск;

2. Магнитная лента.

Дисковые файлы чаще всего используются для хранения резервных копий. Диск с копией может быть расположен на локальном жестком диске на сервере или на удаленном диске в общей сетевой папке. Компонент Database Engine позволяет добавлять новые копии в файл, который уже содержит копии той же или другой базы данных. Когда вы добавляете новый набор копий на существующий носитель, предыдущее содержимое носителя остается нетронутым, и новая копия записывается после последней копии на этом носителе. (Набор копий включает все сохраненные данные объекта, которые вы выбираете для копирования.) По умолчанию компонент Database Engine всегда добавляет новые копии в файлы на диске.

Магнитная лента обычно используется в качестве устройства хранения резервных копий так же, как и дисковый накопитель. Однако при резервном копировании на ленту ленточное устройство должно быть локально подключено к системе. Преимущество ленточных устройств перед дисковыми - простота администрирования и эксплуатации.

Сценарий восстановления — SQL Server - это процесс восстановления данных из одной или нескольких резервных копий и возврата базы данных в исходное состояние. Поддерживаемые сценарии восстановления зависят от модели восстановления базы данных и выпуска SQL Server.

Для большинства сценариев восстановления необходимо применить резервную копию журнала транзакций и позволить ядру СУБД SQL Server запустить процесс восстановления, чтобы перевести базу данных в оперативный режим. Восстановление - это процесс, используемый SQL Server для запуска каждой базы данных в транзакционно согласованном (чистом) состоянии.

В случае аварийного переключения или другого неясного завершения работы базы данных могут оставаться в состоянии, в котором некоторые изменения не были записаны из буферного кеша в файлы данных, и в файлах данных могут быть некоторые изменения, вызванные незавершенными транзакциями. Когда вы запускаете экземпляр SQL Server, выполняется трехэтапное восстановление каждой базы данных на основе самой последней контрольной точки в базе данных:

1. На этапе анализа анализируется журнал транзакций для определения последней контрольной точки и создается таблица грязных страниц (DTP) и таблица активных транзакций (TAT). TGS содержит записи о страницах, которые были грязными при выключении базы данных. TAT содержит записи транзакций, которые все еще были активны, когда база данных была аварийно завершена.
2. На этапе повтора повторяются все зарегистрированные изменения, которые, возможно, не были записаны в файлы данных, когда база данных была закрыта. Минимальный порядковый номер журнала (minLSN), необходимый для успешного восстановления всей базы данных, находится в TGS и отмечает начало операций восстановления, необходимых для всех грязных страниц. На этом этапе компонент SQL Server Database Engine записывает на диск все грязные страницы, принадлежащие зафиксированным транзакциям.
3. На этапе отката выполняется откат ожидающих транзакций, обнаруженных в TAT, чтобы гарантировать сохранение целостности базы данных. После отката база данных переводится в оперативный режим, и больше нельзя применять резервные копии журнала транзакций.

Цель полного восстановления - восстановить всю базу данных. База данных отключена во время периода восстановления. Перед тем, как какая-либо часть базы данных перейдет в оперативный режим, все данные восстанавливаются до точки согласованности, когда все части базы данных находятся в один и тот же момент времени и отсутствуют незавершенные транзакции.

В режиме полного восстановления после восстановления резервных копий данных необходимо восстановить все последующие резервные копии журналов транзакций, а затем и саму базу данных. База данных может быть восстановлена до определенной точки восстановления в одной из этих резервных копий журнала. Этой точкой восстановления может быть указанная дата и время, помеченная транзакция или порядковый номер журнала (LSN).

При восстановлении базы данных, особенно при использовании модели восстановления с полным или неполным протоколированием, используйте одну последовательность восстановления. Последовательность восстановления состоит из одной или нескольких операций восстановления, которые перемещают данные в одну или несколько фаз восстановления.

Глава 2. Создание программного обеспечения для резервного копирование и восстановления базы данных.

При создании программного обеспечения для резервного копирования и восстановления базы данных SQL Server, использовалась среда разработки Microsoft Visual Studio 2019, язык программирования C# на Windows Forms. Структурная схема программы была составлена средствами программного обеспечения Graph Editor и представлена на Рисунке 1.

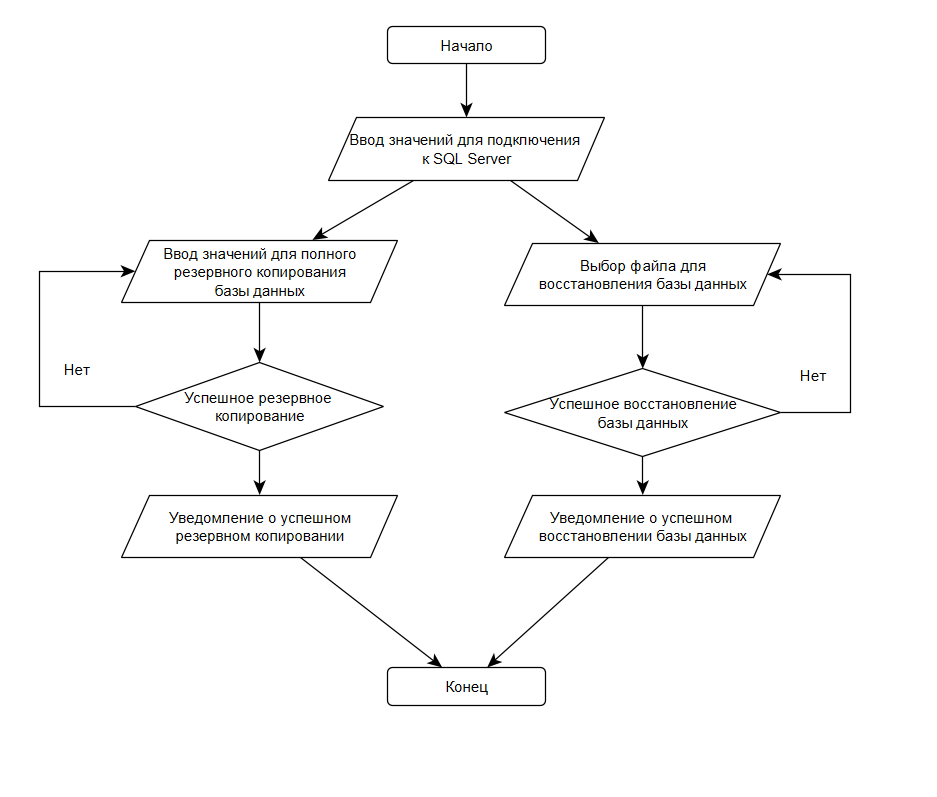


Рисунок 1. Структурная схема программы

При создании программного обеспечения использовались три основных класса:

1. Form.cs
2. BD.cs
3. File

Form.cs – является классом, в котором происходит реализация отклика графических элементов.

public partial class Form1 : Form

{

//Создание объекта класса

BD bD = BD.getInstance();

File file = new File();

public Form1()

{

InitializeComponent();

string[] Parameters = file.ReadFileParametersBD();

textBoxNameServer.Text = Parameters[0];

textBoxNameUser.Text = Parameters[1];

textBoxPass.Text = Parameters[2];

textBoxNameBase.Text = Parameters[3];

//Вызов метода подключения к базе данных, который находится в классе BD

bD.Connection(Parameters[0], Parameters[1], Parameters[2]);

timer1.Interval = 3600000;

timer1.Start();

}

//Обработчик нажатия на кнопку Подключиться

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

{

//Считывание значений с textBoxs

string Server = textBoxNameServer.Text;

string NameUser = textBoxNameUser.Text;

string Pass = textBoxPass.Text;

//Вызов метода подключения к базе данных, который находится в классе BD

bD.Connection(Server, NameUser, Pass);

}

// Обработчик нажатия на кнопку Резервное копирование

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

{

// Считывание значения из textBox

string NameBase = textBoxNameBase.Text;

// Вызов диалогового окна для выбора файла

FolderBrowserDialog folderDlg = new FolderBrowserDialog();

folderDlg.ShowNewFolderButton = true;

DialogResult result = folderDlg.ShowDialog();

if (result == DialogResult.OK)

{

// Присваивание переменной пути до файла

string Directory = folderDlg.SelectedPath;

// Обработчик исходов резервного копирования

int state = bD.Backup(NameBase, Directory);

switch (state)

{

case 0:

MessageBox.Show("Резервное копирование завершено");

break;

case 1:

MessageBox.Show("Ошибка резервного копирования");

break;

case 2:

MessageBox.Show("Вы не подключены к Sql Server");

break;

}

}

}

// Обработчик нажатия на кнопку Выбрать файл

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

{

// Открытие диалогового окна

OpenFileDialog OPF = new OpenFileDialog();

if (OPF.ShowDialog() == DialogResult.OK)

{

// Контроль выбора файла формата .bak

if (OPF.FileName.Substring(OPF.FileName.Length - 3, 3) != "bak")

{

MessageBox.Show("Вы не выбрали .bak файл");

}

else

{

// Отработка метода восстановления базы данных и возвращения исхода

int state = bD.Recovery(OPF.FileName, Path.GetFileNameWithoutExtension(OPF.FileName));

switch (state)

{

case 0:

MessageBox.Show("Восстановление завершено");

break;

case 1:

MessageBox.Show("Ошибка восстановления");

break;

case 2:

MessageBox.Show("Вы не подключены к Sql Server");

break;

}

}

}

}

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

{

string[] Parameters = file.ReadFileParametersBD();

bD.Backup(Parameters[3], Application.StartupPath.ToString());

file.WriteFileIntervalSave(comboBox1.SelectedIndex, DateTime.Now);

}

public void PlanCopy()

{

string[] ParametersInterval = file.ReadFileIntervalSave();

string[] Parameters = file.ReadFileParametersBD();

TimeSpan interval = DateTime.Now.Subtract(DateTime.Parse(ParametersInterval[1]));

switch (int.Parse(ParametersInterval[0]))

{

case 0:

if (interval >= TimeSpan.FromHours(1))

{

bD.Backup(Parameters[3], Application.StartupPath.ToString());

file.WriteFileIntervalSave(0, DateTime.Now);

}

break;

case 1:

if (interval >= TimeSpan.FromHours(3))

{

bD.Backup(Parameters[3], Application.StartupPath.ToString());

file.WriteFileIntervalSave(0, DateTime.Now);

}

break;

case 2:

if (interval >= TimeSpan.FromHours(6))

{

bD.Backup(Parameters[3], Application.StartupPath.ToString());

file.WriteFileIntervalSave(0, DateTime.Now);

}

break;

case 3:

if (interval >= TimeSpan.FromHours(12))

{

bD.Backup(Parameters[3], Application.StartupPath.ToString());

file.WriteFileIntervalSave(0, DateTime.Now);

}

break;

case 4:

if (interval >= TimeSpan.FromDays(1))

{

bD.Backup(Parameters[3], Application.StartupPath.ToString());

file.WriteFileIntervalSave(0, DateTime.Now);

}

break;

case 5:

if (interval >= TimeSpan.FromDays(7))

{

bD.Backup(Parameters[3], Application.StartupPath.ToString());

file.WriteFileIntervalSave(0, DateTime.Now);

}

break;

case 6:

if (interval >= TimeSpan.FromDays(30))

{

bD.Backup(Parameters[3], Application.StartupPath.ToString());

file.WriteFileIntervalSave(0, DateTime.Now);

}

break;

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

PlanCopy();

}

Из выше представленного фрагмента кода, можно сделать вывод, что в классе реализованы 5 основных методов:

1. Обработчик события нажатия на кнопку «Подключиться» (private void buttonConnect\_Click(object sender, EventArgs e))
2. Обработчик события нажатия на кнопку «Резервное копирование». (private void buttonCopy\_Click(object sender, EventArgs e))
3. Обработчик события нажатия на кнопку «Выбрать файл». (private void buttonRecovery\_Click(object sender, EventArgs e))
4. Обработчик события нажатия на кнопку «Сохранить». (private void buttonSaveInterval\_Click(object sender, EventArgs e))
5. Метод периодического срабатывания с интервалом в 1 час, в данном методе происходит автоматическое резервное копирования с учетом установленного интервала. (public void PlanCopy())

BD.cs – является классом взаимодействия команд, созданных на языке программирования C#, с языком SQL, который используется для создания и выполнения запросов, посылаемых в SQL Server.

class BD

{

// Начало создания паттерна одиночки (singlton)

private static BD instance;

private BD()

{ }

public static BD getInstance()

{

if (instance == null)

instance = new BD();

return instance;

}

//Конец создания паттерна

//Объявление поля типа SqlConnection

SqlConnection conn;

//Подключение к базе данных

public void Connection(string datasource, string nameUser, string pass)

{

//Создание строки

string connString = @"Data Source=" + datasource + ";User Id = " + nameUser + "; Password = " + pass + ";";

//Создание объекта

conn = new SqlConnection(connString);

}

//Метод резевного копирования базы данных

public int Backup(string NameBase, string Directory)

{

//Создание строки

string sqlExpression = $"BACKUP DATABASE [{NameBase}] " +

$"TO DISK = N'{fullPath}' " +

$"WITH FORMAT, NAME = N'{fileName} Database Full Backup'"

//Проверка на наличие объекта

if (conn != null)

{

try

{

//Открытие соединения

conn.Open();

//Создание команды

SqlCommand command = new SqlCommand(sqlExpression, conn);

//Исполнение команды

command.ExecuteScalar();

//Закрытие соединения

conn.Close();

return 0;

}

catch

{

conn.Close();

return 1;

}

}

else

{

conn.Close();

return 2;

}

}

//Метод восстановления базы данных

public int Recovery(string FileNamePath, string nameBase)

{

//Создание строки

string sqlExpression = "USE [master] RESTORE DATABASE[" + nameBase + "] " +

"FROM DISK = N'" + FileNamePath +

"' WITH FILE = 1, NOUNLOAD, STATS = 5"; // Проверка на наличие объекта соединения

if (conn != null)

{

try

{

// Открываем соединение

conn.Open();

// Создание объекта команды

SqlCommand command = new SqlCommand(sqlExpression, conn);

// Исполнение команды

command.ExecuteScalar();

// Закрытие соединения

conn.Close();

return 0;

}

catch

{

conn.Close();

return 1;

}

}

else

{

conn.Close();

return 2;

}

}

Анализируя выше представленный фрагмент кода можно сделать вывод, что в классе реализованы три основных метода:

1. Метод соединения с SQL Server

public void Connection(string datasource, string nameUser, string pass)

1. Метод резервного копирования базы данных

public int Backup(string NameBase, string Directory)

1. Метод восстановления базы данных

public int Recovery(string FileNamePath, string FileName)

File.cs – является классом с методами взаимодействия с файлами, в которых записаны параметры подключения к серверу, и установленный интервал с последний временем резервного копирования.

class File

{

public string[] ReadFileParametersBD()

{

StreamReader f = new StreamReader("Parameters.txt");

string[] Parameters = new string[4];

int i = 0;

while (!f.EndOfStream)

{

string s = f.ReadLine();

Parameters[i] = s;

i++;

}

f.Close();

return Parameters;

}

public void WriteFileIntervalSave(int interval, DateTime time)

{

StreamWriter f = new StreamWriter("IntervalTools.txt", false);

f.WriteLine(interval.ToString());

f.WriteLine(time.ToString());

f.Close();

}

public string[] ReadFileIntervalSave()

{

StreamReader f = new StreamReader("IntervalTools.txt");

string[] Parameters = new string[2];

int i = 0;

while (!f.EndOfStream)

{

string s = f.ReadLine();

Parameters[i] = s;

i++;

}

f.Close();

return Parameters;

}

Анализируя данный класс можно выделить три основных метода

1. Метод считывания параметров подключения к серверу.
2. Метод записи интервала резервного копирования и последнего времени резервной копии.
3. Метод считывания интервала резервного копирования и последнего времени резервной копии.

Глава 3. Графическое описание программного обеспечения

Для графической составляющей программного обеспечения использовался механизм Windows Forms. На Рисунке 2 представлено основное окно программного обеспечения.

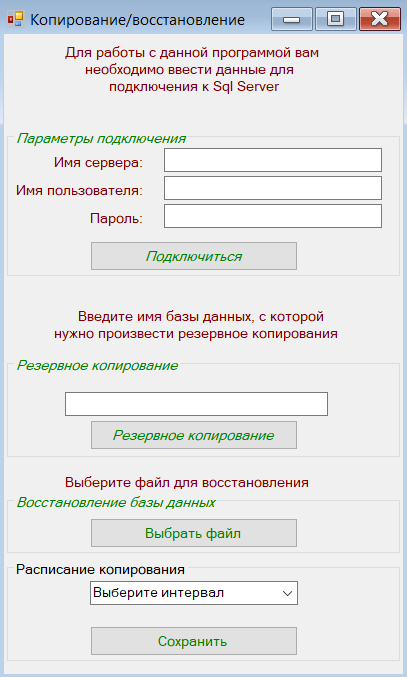


Рисунок 2. Основная форма программы

Данная форма делится на четыре блока:

1. Блок подключения к серверу;
2. Блок резервного копирования;
3. Блок восстановления базы данных;
4. Блок установки расписания копирования БД.

## Глава 4. Тестирование работы программного обеспечения

Создадим тестовую базу данных в SQL Server:

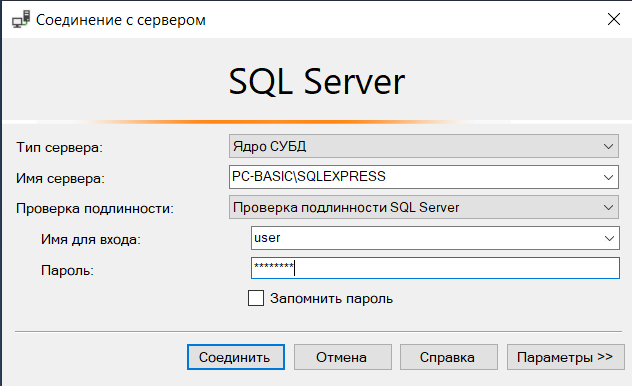


Рисунок 3. Соединение с SQL Server через менеджер управления SQL

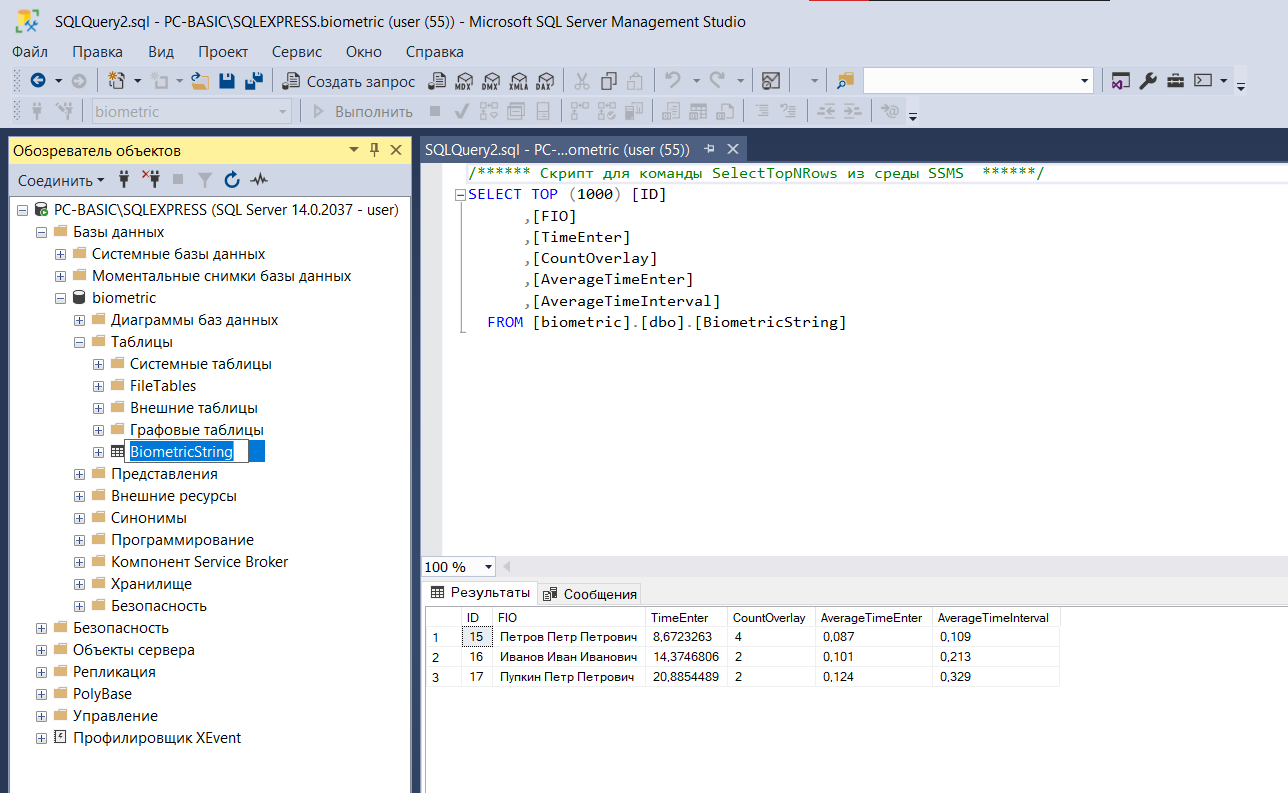


Рисунок 4. Тестовая база данных

Сделаем ее резервную копию через наше программное обеспечение:

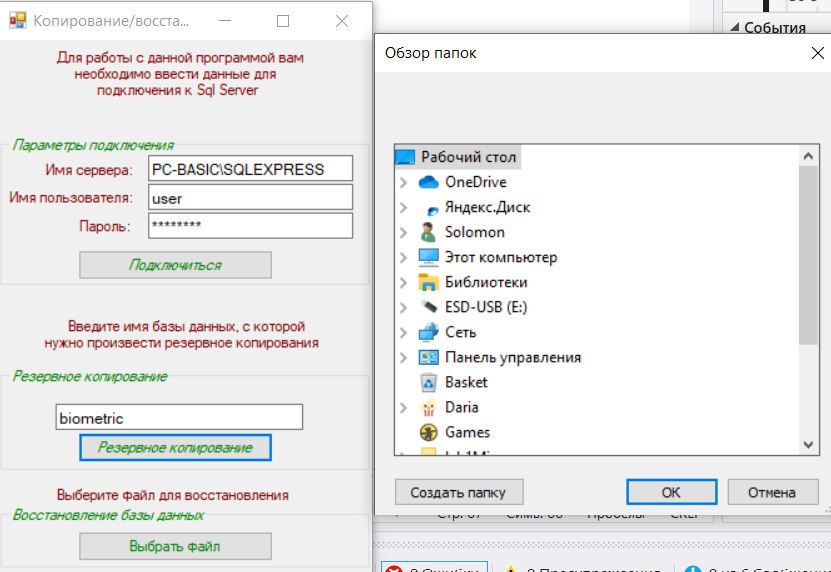


Рисунок 5. Резервное копирование тестовой базы данных

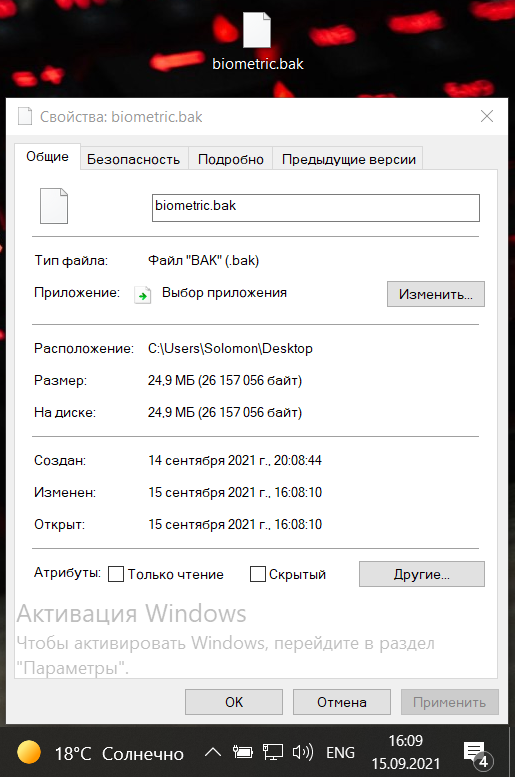


Рисунок 6. Временное подтверждение резервного копирования

Удалим тестовую базу данных из списка баз данных SQL Server:

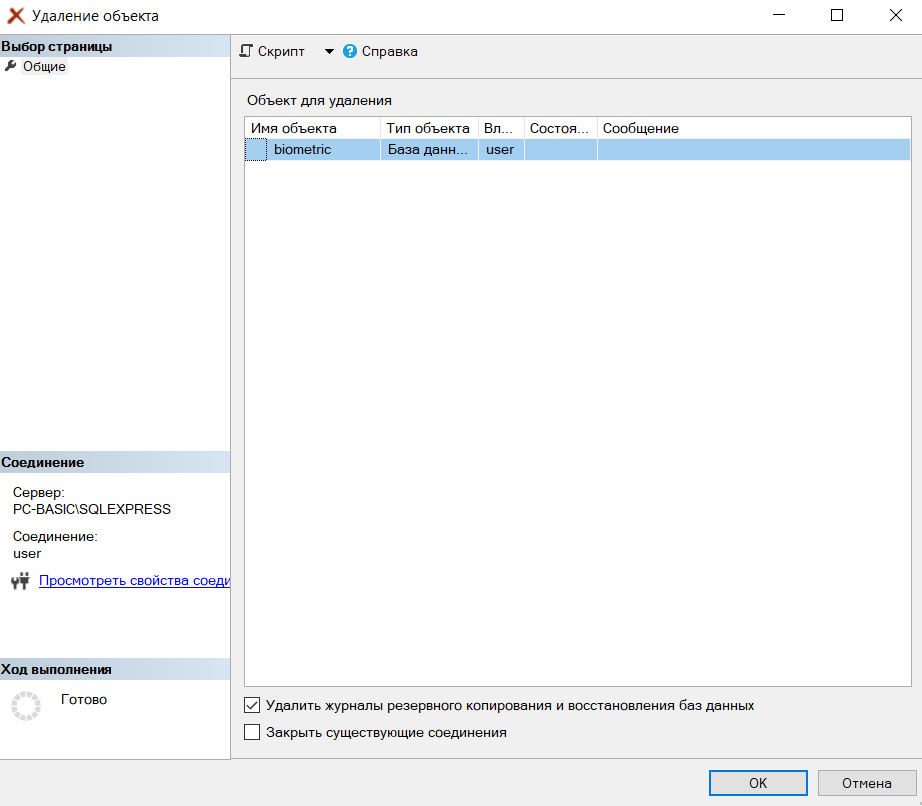


Рисунок 7. Удаление тестовой базы данных

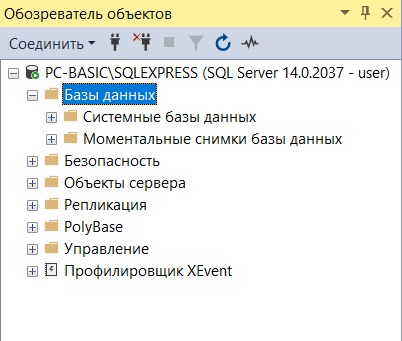


Рисунок 8. Проверка на отсутствие тестовой базы данных

Восстановим тестовую базу данных с помощью нашего программного обеспечения:

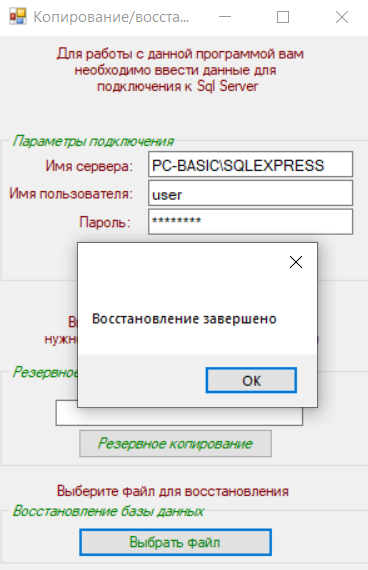


Рисунок 9. Восстановление тестовой базы данных

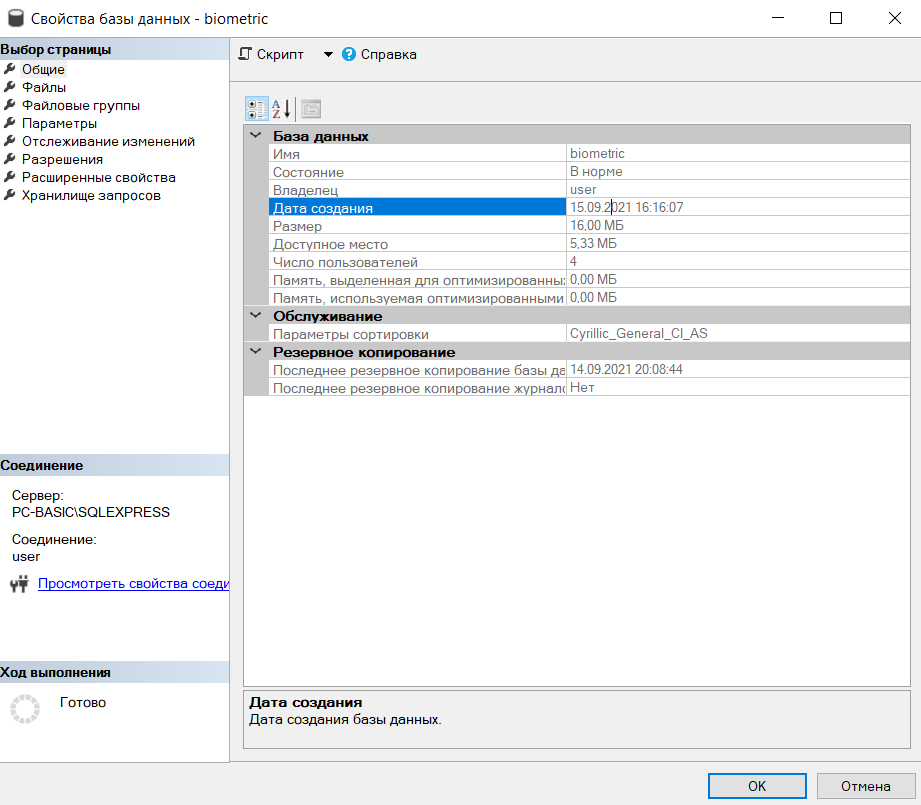


Рисунок 10. Временное подтверждение восстановления тестовой базы данных.

Заключение

В ходе данной работы были рассмотрены основные факторы и алгоритмы, позволяющие провести резервное копирования и восстановление базы данных SQL Server. Построена структурная схема программы, по которой было реализовано программное обеспечение, позволяющее совершать резервное копирование и восстановление базы данных, не использую среду Microsoft SQL Server Management Studio.

Список используемых источников

1. Шилдт, Герберт. C# 4.0: полное руководство. Пер. с англ. — М: ООО «И.Д. Вильямс». Москва, 2011. — 1056 с.
2. Осетрова И.С., Разработка баз данных в MS SQL Server 2014. - СПб: Университет ИТМО, 2016. – 114 с.
3. Шарп Джон Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017. — 848 с.: ил. — (Серия «Библиотека программиста»)