OULUN AMMATTIKORKEAKOULU

Jani Inkala

DATABASE MANAGEMENT TOOL

DATABASE MANAGEMENT TOOL

Jani Inkala
Bachelor's thesis
Spring 2024
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author(s): Jani Inkala
Title of the thesis: Database management tool
Thesis examiner(s): Teemu Korpela
Term and year of thesis completion: Spring 2024 Pages: 18

The objective of this thesis was to create a simple tool to help manage manual data deletion processes done to certain testing databases within Nestix. The development of the said tool was done by the author of this thesis.

This thesis first covers the used tools and the reasons why the author chose to use those tools over other widely used technologies and coding languages. The thesis lightly covers the chosen tools on both technical perspective and on the author's personal perspective.

After having covered the used tools, the next part of the thesis goes into the planning process which explains how UI, functions and database connections were built. The end of the planning section also shows the state of the current tool with all its functionalities.

Ideas for further development section covers a couple of ideas that were taken into consideration during the development process but could not be implemented within the given timeframe. These ideas include ways to further refine deletion processes by including ways to specify on certain conditions on which data is to be deleted.

Finally, a short summary is given on the project and how the end goal of the project was met.

Keywords: T-SQL, C#, .NET

PREFACE

The development and research for the thesis were done by the writer of the thesis. The objective was to create a tool for lessening manual labour on database management within the company's testing servers. The primary aim was to create the basis for accessing the correct databases via GUI and have the option to do delete functions for different data tables. This thesis was supervised by Marko Ukkola from Nestix Oy and Teemu Korpela from OUAS.

Oulu, 8.1.2024
Jani Inkala

CONTENTS

VOCABULARY

SQL = Structured Query Language, which is used to interact with databases

T-SQL = Transact Structured Query Language. A query language which is specific to the Microsoft SQL server product.

GUI = Graphical User Interface

SSMS = SQL Server Management Studio, Tool for managing databases

WPF = Windows Presentation Foundation, open-source graphical user interface library

WinForms = Windows Forms, open-source graphical user interface library

1 INTRODUCTION

This tool is planned to be used in specific case of deleting data from within testing databases and rather than always use a multipurpose database tool like SSMS, it would be easier to use the tool with a smaller scope and slightly more pre-planned use cases (1).

The plan for this application is to simplify different data table emptying processes that are needed to manage the product's database. Building an application that would run prebuilt SQL processes would remove the manual part of having to write these deleting scripts from ground up while also trying to figure out correct order in which the tables need to be emptied.

The main purpose of this application is to manage deleting information from data tables within multiple databases but to do that, there is a need to access the correct databases and building the correct scripts either into the application itself or into each of the databases. The options and the decisions between the chosen and scrapped ideas will be looked at in a later chapter of this thesis.

The chosen coding tool used to create the application introduced in this thesis was Visual Studio Community (2). The chosen coding language was C#, and the application was built within .NET framework. Chosen UI framework for this application was WinForms (3).

2 DEVELOPING THE TOOL

The design for the tool was done according to the needs of the company Nestix, where the author worked during this thesis project. The GUI was designed by the author and the required functionalities were discussed with the supervisor and other testers within Nestix. The development of both GUI and functions were done by the author. Nestix already had different servers build for different versions of its products and thus the procedures and other functions were built around existing server structures.

2.1 Choosing the building tools

The primary objectives for this thesis project were to have a tool that was executable by itself, and it would be capable of running some predetermined SQL scripts within multiple preconstructed databases. The first choice regarding the tools used to build this application was that it would be done by mostly using C# and .NET framework. Another clear choice was the use of T-SQL since it was already widely used within the company. Widely used options for building a GUI that were considered for this project were WinForms and WPF of which WinForms was chosen due to it being more light weight and simpler in both visual and procedural aspects (4).

2.1.1 Why use .NET?

The writer of this thesis landed on using .NET due to personal preferences and after having studied some well-known options for the framework while to be used in this project. The choice was based on basis of .NET being great for windows desktop applications and web applications, having a large user base and a good performance comparatively.

Having a good base for both desktop applications and cloud applications was one of the main criteria while choosing a framework for this thesis. .NET provided a great option for creating an application that could be first created on desktop environment and if needed, it could also be transferred to cloud much easier than other counterparts such as Node.js (5).

One determining factor for using .Net came from large user base which helps this project directly while also enabling the writer of this thesis to improve on a highly demanded programming language framework. Having a large user base meant having access to a huge amount of community supported libraries and thus it enabled the ability to hasten and improve the programming process (6). Having a proof of a large user base also confirmed the need for having good understanding of .NET due to the possibility of having to use it in any possible future projects.

Lastly one major factor taken into consideration was performance. .NET was proven to be faster compared to other counterparts like Node.js in matters such as database access, which was a major part of the functions of the planned application (7).

2.1.2 C# vs other programming languages

The programming languages were reduced in early phases to C#, JavaScript, and Python. After having come to a conclusion of using .NET as a framework for this project, Python and JavaScript were quickly excluded due to incompatibilities between them and .NET framework. However, the choice of using C# was also enforced due to some other reasons other than just compatibility with .NET.

On a personal level having chosen C# provided the author a chance to improve on skill set that is on high demand and has deep connection to services provided by Microsoft (8). Having already worked with C# to some extend yet significantly less than JavaScript gave the author a good chance to improve upon personally less used programming language rather than work with the same language time after time.

2.2 Planning

In the beginning of the planning phase, this project was divided into three categories. These mentioned categories were GUI, required functions to make this a working executable and database along with the data within. These three categories and their planning processes will be further explained in the following chapters.

2.2.1 Graphical User Interface

When considering how the UI would look like, the author of this thesis mostly considered how to make it easily manageable rather than make it look as nice as possible. The original plan for functions for this tool was quite simple and thus a simple and fast UI was thought to suffice quite well. Another important aspect of the tool that was discussed was to make it easily expandable for possible future ideas that could be included in the tool.

As mentioned in an earlier chapter about choosing the building tools, WinForms GUI library was chosen over WPF since priorities in this project were to create a lightweight and simple to use application. Out of the two, WPF would have been the better options if any emerged ideas had included graphically demanding processes. Keeping demanding graphics away from the scope of the project allowed for a use of a more lightweight option which was stated as an important quality for the tool.

Early designs of the UI included a few core parts which were to be implemented as a part of the main functionalities of the application. The first element of these parts were two combo boxes of which the first box would have a list of servers that a user could connect to, and the second box would contain the names of the databases within these servers. The second core component of the project was to have a list of check boxes with the names of the data tables that could be emptied using the tool. The third core components to be included were different buttons for running the SQL commands and procedures in the chosen databases.

2.2.2 Functions behind the interface

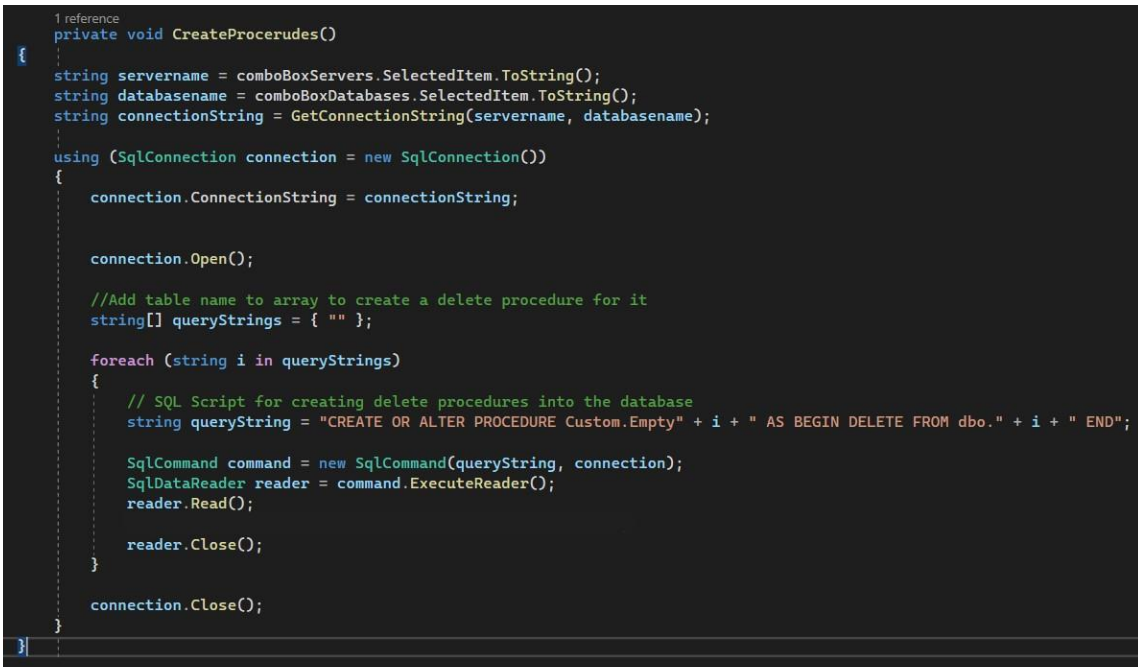
As important as UI is from the user’s perspective, it is still only a medium between the user and the working core of the software. Designing functions to manage interface actions was a key point when coming up with what is needed to make the interface itself work along with making sure that those functions would access the correct databases while using correct SQL commands.

Functions that make the interface itself work consist of functions and objects that are directly on the form and mostly visible to the user as either a part of the form, such as a button or as a visible action that the visible parts trigger. The functions working within the form were mostly implemented into the checkboxes that list the used data tables which could be accessed through the interface.

These mentioned functions were planned to consist of a few parts of which each would deal with a part in the chain that would be required to have a list of desired data table names in correct order. The first part of this chain was to have functions for each checkbox to check other boxes if such action was needed to make sure that if the data of the original table could be deleted without having data from linked tables inhibiting the process. The second part of the process chain was to get the included tables into correct order. By having the tables in correct order in the deleting process could be run smoothly due to any conflicting data having been removed before accessing further parts in the list of data tables.

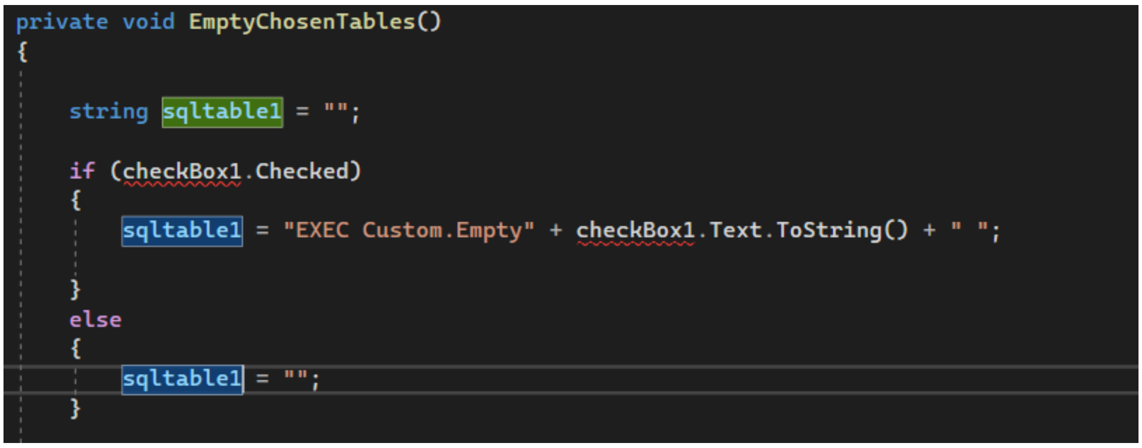
Lastly, a set of functions was needed to have the means to open and close accesses to databases and run the required SQL scripts within them. The opening and closing processes were planned to happen right before and after the actual scripts themselves to lessen the burden on the servers as well as limit the access times to minimal. The plan included having two basic sets of procedures. First a function for creating the full list of procedures into the database would be built (Figure 1.). The procedure creating function was designed to have an array of table names which could then be used to create a loop of SQL commands, each for creating one delete procedure.

The second set of functionalities were designed as a means for accessing the deleting procedures for the chosen tables. The functions itself were designed to first have a list of strings, each representing one of the tables. The value of a single string would be either execution command for the assigned table or an empty string based on whether the checkbox appointed for the same table was checked or not (figure 2.). After having given these strings values, they would be used to build a single string which would then be used as a whole command line for the deleting process (Figure 3.). After having created the full command line to be used in the SQL connection, the function would retrieve information about the server and database names. The names would then be added to their own places within the connection string (Figure 5.). After having constructed the connection string and SQL script, a connection to the server would then be opened and the built script would be executed after which the connection would be closed (Figure 4.).



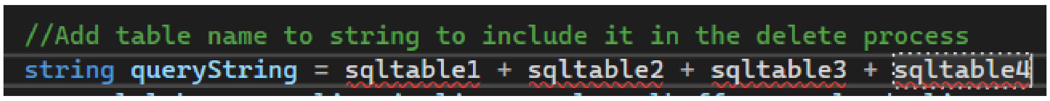
```
    1 reference
    private void CreateProcerudes()
{
    string servername = comboBoxServers.SelectedItem.ToString();
    string databasename = comboBoxDatabases.SelectedItem.ToString();
    string connectionString = GetConnectionString(servername, databasename);

    using (SqlConnection connection = new SqlConnection())
    {
        connection.ConnectionString = connectionString;


        connection.Open();

        //Add table name to array to create a delete procedure for it
        string[] queryStrings = { "" };

        foreach (string i in queryStrings)
        {
            // SQL Script for creating delete procedures into the database
            string queryString = "CREATE OR ALTER PROCEDURE Custom.Empty" + i + " AS BEGIN DELETE FROM dbo." + i + " END";

            SqlCommand command = new SqlCommand(queryString, connection);
            SqlDataReader reader = command.ExecuteReader();
            reader.Read();

            reader.Close();
        }

        connection.Close();
    }
}
```

FIGURE 1. Function for creating required SQL procedures into a database.



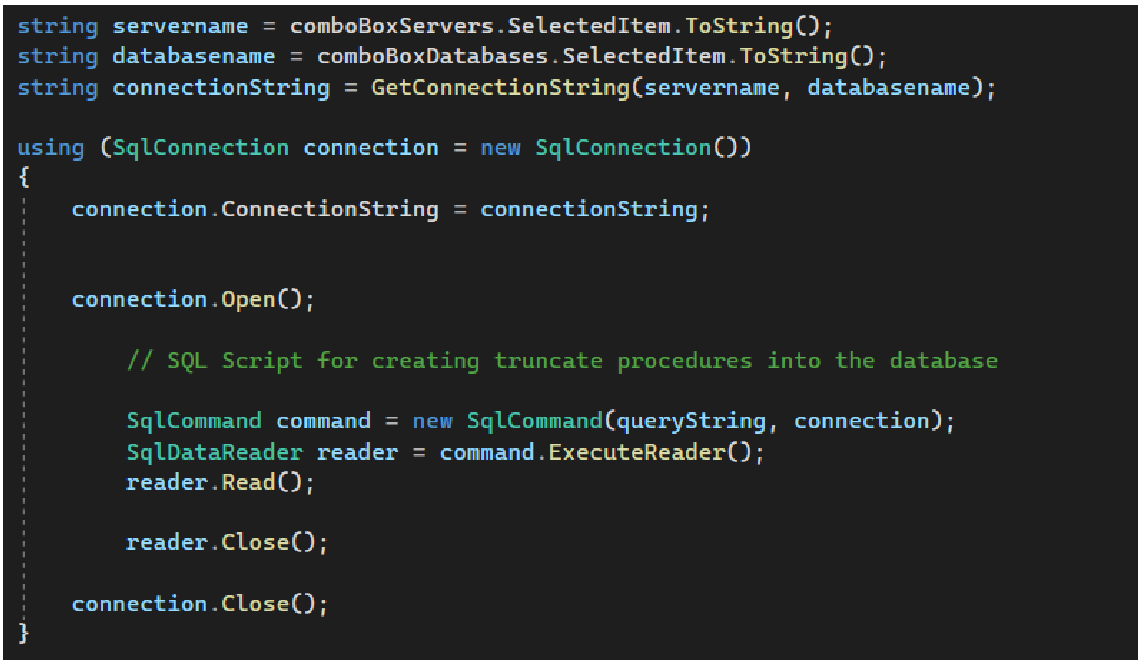
```
private void EmptyChosenTables()
{

    string sqltable1 = "";

    if (checkBox1.Checked)
    {
        sqltable1 = "EXEC Custom.Empty" + checkBox1.Text.ToString() + " ";

    }
    else
    {
        sqltable1 = "";
    }
```

FIGURE 2. First part of the function for deleting data from tables by accessing procedures in the database.



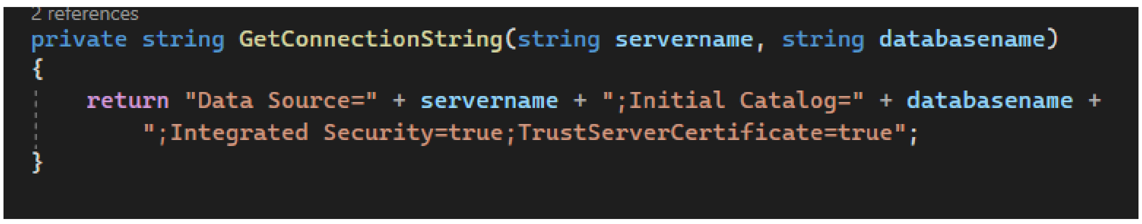
```
//Add table name to string to include it in the delete process
string queryString = sqltable1 + sqltable2 + sqltable3 + sqltable4
```

FIGURE 3. Second part of data deleting function.



```
string servername = comboBoxServers.SelectedItem.ToString();
string databasename = comboBoxDatabases.SelectedItem.ToString();
string connectionString = GetConnectionString(servername, databasename);

using (SqlConnection connection = new SqlConnection())
{
    connection.ConnectionString = connectionString;


    connection.Open();

        // SQL Script for creating truncate procedures into the database

        SqlCommand command = new SqlCommand(queryString, connection);
        SqlDataReader reader = command.ExecuteReader();
        reader.Read();

        reader.Close();

    connection.Close();
}
```

FIGURE 4. Final part of data deleting function.



```
2 references
private string GetConnectionString(string servername, string databasename)
{
    return "Data Source=" + servername + ";Initial Catalog=" + databasename +
        ";Integrated Security=true;TrustServerCertificate=true";
}
```

FIGURE 5. Connection string used for connecting to databases.

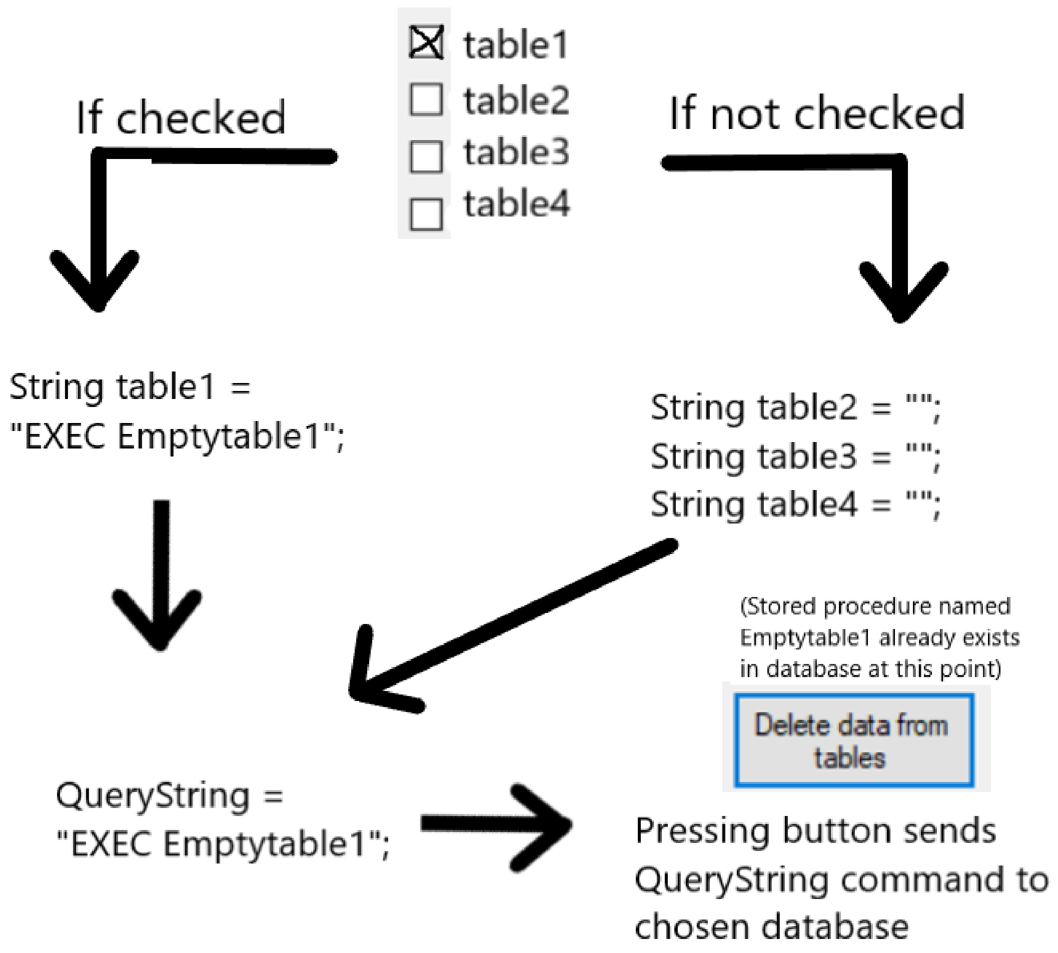


table1
table2
table3
table4
If checked
If not checked
(Stored procedure named Emptytable1 already exists in database at this point)
Delete data from tables
Pressing button sends QueryString command to chosen database

```
String table1 =
"EXEC Emptytable1";
```

```
String table2 = "";
String table3 = "";
String table4 = "";
```

```
QueryString =
"EXEC Emptytable1";
```

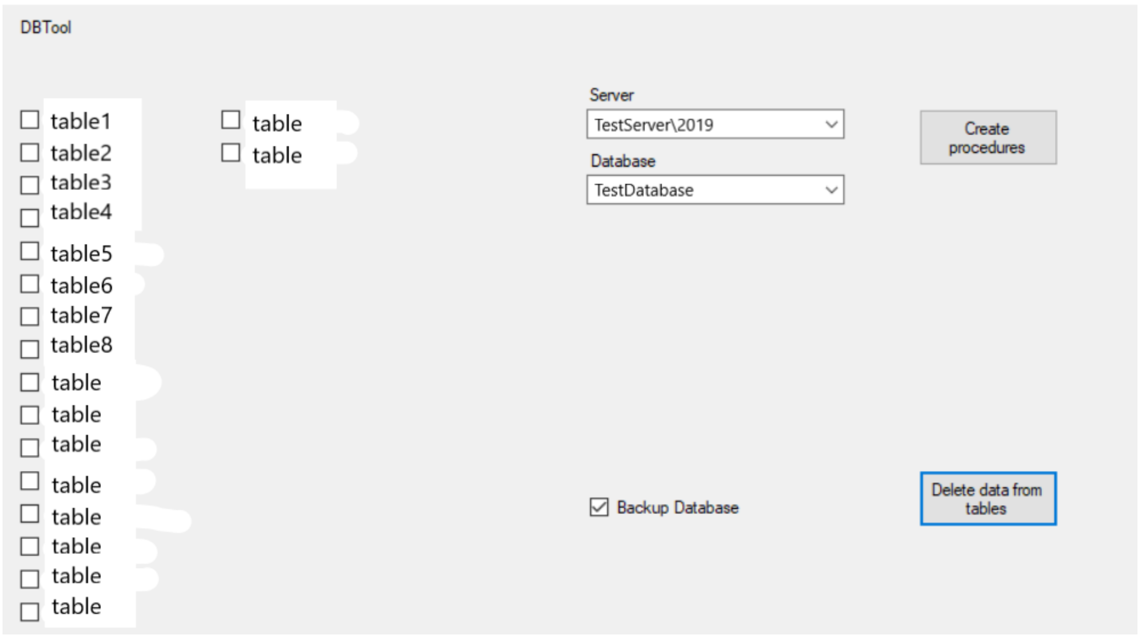
FIGURE 6. Visualisation of the process from checkboxes to pressing the data deletion button.

2.2.3 Data and database

The basic purpose of this executable tool was to be able to manage data within already existing databases. The planned way to do these modifications was to once create pre-planned procedures into the databases where they could be called rather than run the commands straight from within the tool every time. Creating specified procedures into the databases is already used to make customizations to Nestix's product and thus makes it an easy way to create reusable and easily managed functions. Having the SQL procedures in the databases itself makes it easier to transfer the created tool into cloud later if needed.

2.3 The current version of the tool

This chapter presents the state of the created tool as it was at the time of ending the thesis work. Data table names, server names and database names have been hidden from the included figures that are used to show both the code and visual aspects of the tool. A visually humble looking yet decently practical UI (Figure 6.) has been created including a button for creating required processes into the database and another button for handling deleting processes. Currently included data table names have been added alongside their representative checkboxes for each of the tables. Two combo boxes have been added to present either servers or databases that can be connected to.



Server: TestServer\2019
Database: TestDatabase
☑ Backup Database

FIGURE 6. The current UI of the tool without the actual table, server, and database names

Currently working functionalities are the ones that allow for a creation of deletion processes into the chosen databases and functions for using the created procedures to delete data from chosen data tables in correct order. Currently no further ways to make targeted selections or filtration on the accesses data has been implemented.

3 IDEAS FOR FURTHER DEVELOPMENT

As a tool with the planned functions which have been created to empty certain data tables, a solid base for future additions has been created. Having a simple way to access the databases allows the tool to be used for multiple data modifying functions but including such options to the tool would only be sensible in cases where these actions would need to be used in high repetition. This reason makes multipurpose database tools like SSMS better for modifying actions such a specific customizations or permanent changes to the database.

3.1 Filtering data

Current data filtering for the created tool is only based on table names but having options for filtering data based on the data within the tables could be used to increase the use cases of the already built in deleting processes. Having an option for using dates or part names would enable deleting processes within defined categories and thus giving flexibility. Having such filtering possibilities included in the tool would also enable them to be used in the possible future functions.

3.2 Changing old dates for testing parts

One question which had risen during the development process was whether it would be possible to change dates for old testing parts so they could be repurposed for certain testing cases. This option would require some investigation into checks in the product that look into certain dates of the parts and the related tables of those checks. On behalf of the tool, it would only require some small additions to the process creation functions while also creating a way to filter the desired parts whether it would be through the dates themselves or some other variable. Lastly including an object to the form as an input for data given by the user would be required to give a specific data value.

3.3 Creating a test part into a correct state of process line

One larger implementation that was discussed loosely during the development process was the possibility to create a test part into different states within the process line. This option is one that would require not only vast studying into requirements on multiple phases of the required data of the created part in both database and files. Keeping a part's data integrity would be a handful and to implement the required choices to the interface would also be a time-consuming task. Also, when considering the original aim of this tool, this kind of an option is starting to increase the scope of the project to insensible lengths.

4 CONCLUSION

Creating an executable for use cases such as the ones presented in the thesis is quite a simple concept at first, but including databases with complicated data structure gives the project factors that can easily be left unnoticed in the planning process. Most of such problems that emerged during the development process took their toll on the design options. The options that were landed on, however made a good solution as a whole.

While considering the planned requirements and timeframe for this thesis work, the minimum requirements were met adequately. Having a tool that has the required properties to empty certain chosen data tables gives a possibility to quickly run desired deletion processes against multiple databases on different servers. The value of the benefits given by the tool have not been completely tested or evaluated with the tool's current capabilities as of the closing of this thesis work.

The tool at its current form is a good option for deleting tables of their data and the means to include either new tables or procedures quickly have been taken into consideration. The only part requiring extra manual work is to take the database structure into consideration when adding tables due to having to consider the linked tables and the order in which the processes would be executed.

REFERENCES

1. ITenterprice. 2023. What is bespoke software, and why should you go bespoke? Search date 17.11.2023. https://itenterprise.co.uk/bespoke-software-go-bespoke/

2. Microsoft. 2023. Visual Studio Community (Version 17.8). Search date 22.11.2023. https://visualstudio.microsoft.com/vs/community/

3. Microsoft. 2023. Desktop Guide (Windows Forms .NET). Search date 26.11.2023. https://learn.microsoft.com/en-us/dotnet/desktop/winforms/get-started/create-app-visual-studio?view=netdesktop-8.0

4. ByteHide. 2023. WPF vs WinForms – Which One is Right for Your Project? Search date 28.11.2023. https://www.bytehide.com/blog/wpf-vs-winforms

5. TechMagic. 2023. Node.js vs .NET: What to Choose in 2023. Search date 3.12.2023. https://www.techmagic.co/blog/node-js-vs-net-what-to-choose/

6. Hackr.io. 2023. What is .NET? Why You Should Use This Framework. Search date 4.12.2023. https://hackr.io/blog/what-is-net

7. Microsoft. 2023. Why Choose .NET? Search date 2.12.2023. https://dotnet.microsoft.com/en-us/platform/why-choose-dotnet

8. Ideamotive. 2021. C# vs JavaScript: Which Programming Language Is Better for Your Needs? Search date 4.12.2023. https://www.ideamotive.co/blog/c-sharp-vs-javascript