Universität Ulm | 89069 Ulm | Germany

Faculty of Engineering, Computer Science and Psychology
Databases and Information Systems Department

Conceptualization and Realization of a Database Migration Path for an International and mHealth Tinnitus Database

Bachelor’s Thesis at Universität Ulm

Submitted by:
Hiba Bouzaida
hiba.bouzaida@uni-ulm.de

Reviewer:
Prof. Dr. Manfred Reichert

Supervisor:
Dr. Rüdiger Pryss

2019

Version from January 13, 2020

Composition: PDF-$\LaTeX\,2_\varepsilon$

Abstract

The mobile crowdsensing platform TrackYourTinnitus (TYT) was created to monitor and visualize fluctuations of tinnitus perception by affected individuals using smart devices. The platform aims to gather data of tinnitus patients for research purposes and to help those affected to better understand the fluctuations of tinnitus perception. Users have to answer specific questionnaires to assess tinnitus perception and tinnitus-related parameters during their daily routine. The gathered data from the questionnaires are stored in the MariaDB database running on the back-end of the application.

In the future, the data of TrackYourTinnitus will be merged with clinical databases to broaden the researches related to the tinnitus symptom. Consequently, the amount of data stored in the relational database will notably increase. Additionally, MRI scans will be joined to patients' data to allow a better overview of the tinnitus development for individuals.

For this purpose, it is considered to look for an alternative system for hosting the TYT database, since the current database running on MariaDB does not deliver the performance required. This work attempts to transfer the TrackYourTinnitus database from MariaDB to SQL Server. The system migration aims to ensure smooth database operations when dealing with data on a large scale as well as to benefit from the advanced features of T-SQL, the query language used by SQL Server.

Acknowledgment

I would like to express my enormous gratitude to all those who contributed to the success of this bachelor thesis through their professional and personal support.

Special thanks go to my supervisor Dr. Rüdiger Pryss, who allowed me to perform this work under the best possible conditions. I am infinitely grateful to him for sharing with me his knowledge and for his support with this bachelor thesis.

I am also indebted to the endless support of my parents for showing their trust in me and supporting me to pursue my dream towards higher education at a place far from home.

I would like as well to thank my friends who have been very helpful in revising my thesis.

Contents

1 Introduction

Tinnitus involves the sensation of hearing noises when no external sound is present. Each person affected by tinnitus has a different perception of the noises in their ear and these may be temporal or permanent. Around 15 to 20 percent of people are suffering from it [1]. Another study revealed that between 5.1 and 42.7 percent of the world population experience tinnitus at least once during their lifetime. Therefore, it should be emphasized that tinnitus is a serious disease. Despite its prevalence, the disorder is still difficult to treat and available treatments are only effective for patient subgroups. Moreover, it is required to assess several symptoms (such as loudness, fluctuations of tinnitus perception, stress-level, etc.) and to retrieve a large amount of patients' data to enable the diagnosis and treatment of tinnitus. For this purpose, the mobile crowdsensing platform TrackYourTinnitus (TYT) was developed in the context of a large tinnitus database project by a multidisciplinary research team consisting of psychologists, physicians as well as computer scientists from the universities Ulm and Regensburg [2, 3]. The latter contributes to prospectively monitor symptom variability under real-life conditions during a patient's day using smart mobile devices. The TYT platform consists of a website implemented using the Laravel framework, two mobile applications (for iOS and Android) as well as a relational database as the central repository for the collected data [2]. Users are asked to answer specific questionnaires to assess tinnitus perception and tinnitus-related parameters during their daily routine, while the environmental sound level is recorded. The gathered information from these questionnaires is then transferred to the TYT back-end and stored in the MariaDB database running in the background of the system. This allows researchers to evaluate the data collected as well as the user to better understand the disease by visualizing the fluctuations of their tinnitus perception [3].

In the future, the data of the TrackYourTinnitus platform will be merged with clinical databases to improve the diagnosis and therapeutic treatment. As a result, the amount of data stored in the MariaDB database will notably increase. Moreover, it is considered to upload and store MRI scans for each patient for a better assessment of tinnitus variation.

1.1 Problem Statement

The database platform is the core of the dynamic content. It does not only retrieve personal and general information related to patients, but also stores and secures all these data. Nevertheless, the current performance of the database, which is hosted on MariaDB, is not satisfying and is impacting the ability to deliver the best results. For this reason, it is considered to choose a suitable alternative hosting system for the TYT database, which will become the core of the dynamic content moving forward. Furthermore, the Laravel framework, which is used for interacting with the database, supports only four databases, namely MySQL/MariaDB, SQL Server, PostgreSQL, and SQLite [4]. Therefore, the choice of an alternative system is limited. This thesis attempts to migrate the TYT database from MariaDB to Microsft SQL Server, which provides high scalability, performance, as well as security. Yet, migrating very large databases from one system into another is a major challenge. It is important to consider several issues such as data loss, the security of the data as well as technical aspects of the technologies involved.

Moreover, it is considered to extend the TrackYourTinnitus database with MRI scans, which have to be associated with patients and stored as BLOB files in the database (see section 4.3). Accordingly, it is required to test the functionality as well as the performance of the system considering this feature.

1.2 Objective

This thesis is considered as a preliminary attempt to extend the TYT database with unstructured data and to migrate the database from MariaDB to SQL Server. The functionality and performance of both databases need to be compared in order to reveal which system is more suitable to host the data of the TYT platform. The database migration aims to improve the performance of the system and to ensure smooth database operations when dealing with data on a large scale. This will not only provide enhanced user interaction and faster data retrieval but will also allow a better assessment of the data collected for research purposes.

1.3 Outline

This thesis consists of six main chapters. Chapter 2 intends to give an overview of the work related to the field of supporting patients affected by tinnitus by examining three different works more closely. Chapter 3 serves as a baseline description of the several terms and definitions used within this work. Here, terms such as tinnitus, mobile crowdsensing, medical imaging as well the DICOM are explained in detail to assist in a better understanding of the following chapters. This is followed by an overview of the architecture of the TrackYourTinnitus platform is given in section 3.3.

The main focus of chapter 4 is the implementation. First, the requirements are defined and examined in section 4.1 based on the problem statement that has been described in section 1.1. Then, the used database systems, as well as the migration tool, are presented in section 4.2. Subsequently, the possible solutions are discussed and a suitable alternative will be chosen to reach the aim of this work. Lastly, individual implementation steps will be explained and documented in detail in section 4.3.

Afterwards, an empirical evaluation of the chosen methods and techniques used within the implementation is performed in chapter 5. In this, both database hosting systems, the old and the new one, are compared with each other in section 5.1. Then, the results will be discussed considering their upsides and downsides in section 5.2.

Finally, chapter 6 summarizes the work presented. All the important results are illustrated and summarized in section 6.1. The chapter ends with an outlook on possible future expansions based on the results gained in this thesis in section 6.2.

2 Related Work

In this chapter, three works that deal with similar topics are discussed. The first thesis is about the initial realization of the TYT platform, whereas the second one investigates the treatment dropouts of tinnitus patients. The third work compares the TYT platform with two other methods to determine which one is more adequate to recruit tinnitus samples.

2.1 Conception and Technical Realization of a Mobile Framework for the Support of Tinnitus Patients

Back in 2014, the traditional pen and paper method was used to gather data of tinnitus affected patients to examine the fluctuations of tinnitus perception. Thus, an exact recording of the temporal course of the tinnitus was not possible. Besides, it was difficult to relate the fluctuations to other activities, noise level as well as stressful situations in the everyday life of a tinnitus patient. For this purpose, the TrackYourTinnitus (TYT) platform was implemented by Jochen Herrmann within the scope of his diploma thesis [5]. The framework serves to monitor and visualize these fluctuations using smart devices. This is not only used for research purposes but also helps those affected to better understand the fluctuations of tinnitus perception. The platform comprises a website as well as two mobile applications, one is working on Android devices while the other on iOS operating systems. Both mobile apps are native and can be directly downloaded respectively from the Google Play Store or Apple Store [5].

The website enables users to create an account, fill out statistical questionnaires, display results, as well as obtain detailed information about the project. It also offers administra-

tors the possibility to manage statistical questionnaires, users and groups.
However, monitoring the fluctuations of the tinnitus perception is only implemented for mobile apps. To use the app, the user is first required to log in or to register if he does not have an account yet. The registration can be done either on the website or in the app. After a successful login, the current statistical questionnaires are automatically downloaded. If the user has already answered these questionnaires on the website, the corresponding answers are also downloaded. After that, the user can check the predefined notification settings. Then, the questionnaire for monitoring the fluctuations of the tinnitus perception will be displayed and can be filled out. After saving the answers, the user can check the results through the main menu, where he can also access different areas, such as the notification settings [5].

The TrackYourTinnitus Apps are configured to send notifications to users to remind them to fill out the questionnaire again. The user can define fixed times for those reminders through the notification settings. Otherwise, they will be generated at irregular intervals following the "experience sampling" method [5].

The system architecture of the TrackYourTinnitus platform is described more precisely in section 3.3.

2.2 Studying the Potential of Multi-Target Classification on Patient Screening Data to Predict Dropout Cases

The treatment of tinnitus patients is based on screening data, which involves a medical history, recording of assessments as well as medical tests. The screening requires the acquisition of a large amount of information about several factors such as morbidities, the impact of tinnitus on the quality of life, effects on hearing, etc. [6] The objective behind it, is to design personalized treatment for this illness. However, some patients give up their treatment before completion. For this purpose, a study was conducted to investigate dropout cases and to predict patients that will interrupt the treatment employing multi-target classification on the screening data. This study also attempted to identify questionnaire variables that specify the relationship between tinnitus loudness (the loudness of the sound heard by a patient) and treatment interruption. The results have shown that only two questionnaires can serve to collect early indicators of the likelihood that a patient will abandon the treatment, namely the Tinnitus Severity (TS) and the tinnitus Sample Case History Questionnaire (TSCHQ). The contribution of the other questionnaires was minor. Moreover, it was confirmed that tinnitus loudness and annoyance are not related. As a result, indicators of annoyance may also cause patients to interrupt the treatment. Nevertheless, a larger number of samples is needed to validate these findings [6].

2.3 Outpatient Tinnitus Clinic, Self-Help Web Platform, or Mobile Application to Recruit Tinnitus Study Samples?

The paper represents a study that investigates how tinnitus samples are recruited by three different methods: the TrackYourTinnitus mobile crowdsensing platform, the crowdsourcing platform Tinnitus Talk as well as contacting an outpatient tinnitus clinic (Tinnitus Center Regensburg). The three samples were compared regarding age, gender, and duration of tinnitus. This study aimed to determine whether newer technologies, such as crowdsourcing and crowdsensing, offer the possibility to reach individuals with tinnitus that are different from the ones directly contacting an outpatient clinic [7]. The results revealed that individuals younger than 44 years used more frequently the crowdsensing platform TYT, while individuals aged 55 or older mostly used the crowdsourcing platform. In other words, crowdsensing is suited for recruiting younger individuals, yet older ones are better reached through crowdsensing. Another interesting fact was that the dominant gender across all three samples was male. The crowdsourcing platform Tinnitus Talk had the highest percentage of female individuals, while the crowdsensing platform TYT had the lowest. As a result, it is reasonable to assume that crowdsourcing might be more adequate to recruit women than crowdsensing. Considering the duration of tinnitus, it was revealed that acute and subacute tinnitus are more frequent among the users of crowdsourcing and crowdsensing platforms than those of an outpatient clinic [7].

3 Fundamentals

This chapter serves as a baseline description of the different terms used throughout this work and gives an overview of the architecture of the TrackYourTinnitus Database.

3.1 Tinnitus

Tinnitus is the perception by an individual of a ringing sound or noise in the ears or head, which is not originating from an external source. Each person affected by tinnitus has a different perception of the noises in the ear. Common descriptions are that it is a hiss, whistle, whirr or buzz [8]. The pitch may vary from a low roar to a high squeal, and it can be heard in one or both ears. Tinnitus may be present all the time or intermittently [1].

There are two main types: subjective and objective tinnitus [8].

- **Subjective:** This is by far the most common type of tinnitus. It can be heard only by the affected person. Subjective tinnitus can be caused by ear problems in the outer, middle or inner ear. It may also occur due to problems with the auditory nerves or with the part of the brain associated with hearing.
- **Objective:** This can be heard by somebody else examining the affected person, however, it is very uncommon. It can be caused by different physical effects, such as spasm of the tiny muscles in the middle ear, abnormalities in the blood vessels or increased blood flow to the ear.

Anyone can experience tinnitus, but typical factors causing the development of tinnitus include stress, sudden hearing loss, exposure to loud noise, circulatory problems involving the small blood vessels in the inner ear, high blood pressure, etc. [1] The illness can

significantly affect the quality of life if it is persistent. This is due to patients frequently complain about sleep problems, trouble concentrating, memory problems, depression or anxiety [1].
According to a representative study of "Deutsche Tinnitus-Liga e.V." (DTL) in 1999, around three million Germans were affected by tinnitus, of which approximately 1.5 million classified the magnitude of their symptoms as ranging from medium to unbearable [9].

3.2 Mobile Crowdsensing

Mobile Crowdsensing (MCS) is a technology that enables a large number of people to share data and extract information to measure and map phenomena of common interest through mobile devices such as smartphones, tablet computers and wearables. Mobile crowdsensing platforms aim to operate with users and individuals, assign tasks to reliable users, collect the required data and manage it according to the purpose intended for the application. As a result, the use of mobile devices does not only facilitate gathering data, but it also reduces time and costs [10].
The technology has gained popularity in recent years and has become an appealing method for data collection. Many technology companies across the world employ Mobile Crowdsensing to improve the services they provide based on the collected data. Some of the notable examples being Facebook, Google and Uber [11].

With the rise of smartphone sensing, Mobile Crowdsensing has become a promising paradigm for many fields, including the healthcare field. For example, individuals can wear wireless sensors to measure heart rate and blood pressure, and they can transfer their information to the users' equipment. The gathered data can then be used by mobile crowdsensing platforms to perform large-scale studies, i.e. in the healthcare field [10].
In the context of the TYT platform, mobile crowdsensing is used for data collection scenarios related to questions on chronic disorders in order to reveal more detailed information about the development of the illness over time [12].

3.2.1 Medical Imaging

Medical imaging refers to the techniques and processes used to create visual representations of the interior of the human body for clinical analysis and medical interventions. The aim of this procedure is to reveal internal structures hidden by the skin and bones to diagnose and treat diseases. It includes various radiological imaging technologies such as X-ray radiography, Magnetic resonance imaging (MRI), ultrasound, endoscopy, thermography, medical photography, etc. [13]

Medical imaging techniques produce large amounts of data that need to be stored for archiving and telemedicine applications.

3.2.2 DICOM

DICOM (Digital Imaging and Communications in Medicine) is the international standard to transmit, store, retrieve, print, process and display medical imaging information. Besides being free to download and to use, the standard is also actively developed and maintained to meet the evolving technology needs of medical imaging. Many devices comply with the DICOM format, including scanners, servers, 3D printers, etc. Consequently, it has been widely adopted by hospitals, clinics and imaging centres [14]. DICOM files are stored lossy or lossless in TIFF or JPEG format and are then embedded in a file (.dcm). Multiple data types can be contained into these files, including scans (MRI, ultrasound, findings, X-ray, etc.), segmentation, patient data and even 3D data. A compatible program or framework is required to manage or display .dcm extensions. Otherwise, it will not be possible to access the stored content [15].
The copyright of the DICOM Standard is held by The National Electrical Manufacturers Association (NEMA) and administered by the Medical Imaging Technology Association (MITA), which is a division of NEMA and also the secretariat of DICOM [14].

3.2.3 Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is a medical imaging technology used since the early 1980s [13]. It employs a strong magnetic field and radio waves to generate tomographic images of the organs in the human body. MRI produces three-dimensional detailed images that serve to detect and diagnose diseases, as well as to monitor treatment. This technique does not involve the harmful ionizing radiation of X-rays, which is why it is often used to image the brain, spinal cord and nerves, as well as knee and shoulder injuries [16].

3.3 System Architecture

The TYT mobile crowdsensing platform comprises a website, two mobile applications (for IOS and Android) and a back-end. The website[1] was developed with the PHP-Framework Laravel (subsection 3.3.2). Both mobile applications are developed natively, which means that the iOS app was implemented using Objective C and the Android app using Java. However, the server-side back-end is the core of the project and also of this work. The server uses Linux as an operating system, Apache as a web server and PHP as a scripting language (subsection 3.3.1). The communication between the web application and the mobile applications is enabled via a REST-like JSON API[2] [5, 3].

The back-end uses MariaDB, a derivative of MySQL, as a relational database hosting system (subsection 4.2.2). All data visualized or collected by the platform such as users, study information, questionnaires as well as answers are stored in the database. Accordingly, users have access to consistent and uniform datasets, regardless of the device they are using while accessing the platform.

[1] `https://www.trackyourtinnitus.org/`

[2] REST (Representational State Transfer) is an architecture style that defines a set of constraints in order to describe the web architecture [17].

3.3.1 PHP

PHP is an open-source server-side scripting language developed in 1995. Originally, PHP was an acronym for "Personal Home Page Tools", but now it stands for "PHP: Hypertext Preprocessor". PHP derives from Perl and is especially suited for web development. Today, it is also used for command-line scripting and coding applications. PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management system and web frameworks. The current version (PHP 7.4.0) was released in November 2019 [18].

3.3.2 Laravel

Laravel is a free, open-source PHP framework for the development of RESTful web services that follows the MVC (Model-View-Controller) architectural pattern. The first version was released in June 2011 [19]. Laravel provides a unified and well-designed experience for web developers. The framework supports four databases, namely MySQL, PostgreSQL, SQLite and SQL Server [4].

4 Methods

This chapter represents the main part of the thesis. First, the proposed changes to the model set out in chapter 1 are defined and examined (section 4.1). Then, the used technologies and tools are illustrated and described properly (section 4.2). This ensures a better understanding of the decisions taken in the last section of this chapter. Afterwards, a set of solutions is discussed, and a suitable alternative will be chosen. Finally, individual implementation steps are explained in detail (section 4.3).

4.1 Requirements

In each development process, requirements are set up to get an overview of the whole work and allow for better planning and scheduling from the beginning. In this work, the requirements are divided into two parts as shown below:

4.1.1 Upload and Store MRI Scans into the Database

It is required to associate MRI scans to patients by uploading and storing them as files in a designated table in the database. This will permit a better overview of the tinnitus symptom development per patient. The system should be reliable to handle large files of a size greater than 10 MB as well as a massive amount of file uploads due to the large number of users. Moreover, the files should be stored in their original format, so that they can later be processed correctly.

4.1.2 Data Migration from MariaDB to SQL Server

The TrackYourTinnitus platform currently runs on a MariaDB. However, the number of patients and data sets is rapidly increasing over time. Currently, the relational database contains more than 200.000 data sets across 33 tables. Due to the back-end implementation using Laravel (subsection 3.3.2), the alternatives for the database system are limited. For these reasons, it is considered to migrate the current database to Microsoft SQL Server. The database migration aims to increase the performance by optimizing query run times and enabling the storage of a large amount of data. Hence, the resulting system will demonstrate whether the chosen database hosting system is advantageous or not.

4.2 Technologies

This section describes the technologies that were used to develop the existing TYT database project, as well as the tools that were deployed during the analysis and the subsequent implementation.

4.2.1 MySQL

MySQL is an open-source relational database management system (RDBMS) released in 1995. It is based on Structured Query Language (SQL)[1] and developed by Oracle Corporation. The default version uses InnoDB as a storage engine. However, the MySQL storage engine architecture is pluggable, allowing for a specialized storage engine to be used. MySQL belongs to the world's most widely used relational database management systems and supports several platforms like Windows, Mac OS, Solaris, FreeBSD, Linux, and others [21].

[1]SQL is a standardized programming language used by database administrators, as well as other IT-experts to manage databases and query the data stored into them[20].

4.2.2 MariaDB

MariaDB is a world-wide popular open-source relational database created by the original developers of MySQL in 2009. It is the most prominent alternative to MySQL and a favorite among industry giants like Wikipedia, Google, Arch Linux and RedHat [21, 22].

On the one hand, MariaDB is not only highly compatible with MySQL, but it also provides several improvements over running the standard MySQL instance, such as advanced storage engine options, performance improvements, faster caching, etc. Besides, the database is optimized for performance and offers high availability and security, which makes it an enhanced replacement for MySQL [23].

On the other hand, MariaDB is identified as a clone of MySQL despite its various advantages. MariaDB is entirely open-source and free, while MySQL uses a dual licensing to keep its Enterprise edition specific features proprietary. As a result, some functions of the MySQL Entreprise edition are missing in MariaDB due to the closed source. Furthermore, MariaDB originally forked from MySQL 5.5, which means that new features and bug fixes developed for standard MySQL after version 5.5 are not integrated into MariaDB. To solve this, MariaDB conducts monthly merges of the MySQL source code to ensure compatibility as well as feature and bug-fix adoption [22].

4.2.3 Microsoft SQL Server

Microsoft SQL Server (MSSQL) is a relational database management system developed by Microsoft back in 1989. It was originally released exclusively for Windows, but it was made available on Linux in 2016 and it can be run on Mac OS X via Docker. The current version is Microsoft SQL Server 2019, released in November 2019 [20]. Like other RDBMS software, Microsoft SQL Server is based on SQL. However, it is tied to an implementation from Microsoft called Transact-SQL (T-SQL) that has additional proprietary programming extensions to the standard language.

The core component of MSSQL is the SQL Server Database Engine, which is responsible for data storage, processing and security. Some of the database engine features include storing data in instance tables, importing XML data, BLOB data management (section 4.3), transaction logs, data compression, etc. Besides, SQL Server relies fully on the graphical user interface provided by SQL Server Management Studio (SSMS), which facilitates the interaction with databases. Moreover, SQL Server offers many tools that make database development fast and simple, such as Integration Services, a tool for importing and exporting database schemas in an easy way [24].

SQL Server is available in four primary editions: Enterprise, Standard, Developer and Express. Each one of them has different bundled services and tools. However, only the Developer and the Express edition are available free of charge. The first one is used for educational and testing purposes, while the second one is rather used for small databases with sizes up to 10 GB of disk storage capacity [20].

SQL Server has been improved with several features and functions with every release, enhancing the performance, reliability as well as the security of data workloads. According to the National Institute of Standards and Technology's (NIST's), SQL Server has been the most secure database over the last eight years [25]. It supports advanced security technologies, such as Always Encrypted, which permits updating encrypted data without previous decryption, Row-Level Security, which enables data access to be controlled at the row level in database tables, as well as dynamic data masking, which checks user privileges before granting access to the data [20].

T-SQL

Transact-SQL (T-SQL) is an extension of SQL developed by Sybase and Microsoft. It is used to query the database for information, filter and sort records, and join tables. T-SQL extends the standard language with several features, such as transaction control, exception and error handling, row processing as well as declared variables [26].

Integration Services

Microsoft SQL Server Integration Services (SSIS) is a platform for building high performance data integration and data transformations solutions. Integration Services is an ETL tool (Extract, Transform and Load). In other words, it has the ability to extract and transform data from a wide variety of sources such as XML data files, flat files and relational data sources and load it into destination databases or files [27].

4.2.4 ESF Database Migration Toolkit

ESF Database Migration Toolkit is a comprehensive application that allows converting between multiple database formats in a fast and easy way, and without writing any scripts. A wide range of databases is supported including Oracle, MySQL, MariaDB, SQL Server, SQLite, PostgreSQL, IBM DB2, Microsoft Access, etc. The software offers a user-friendly graphical user interface to connect to the source database, select tables and views, then convert them to the target system. The tool supports most database objects except procedures, functions and triggers. Besides, views will be converted into tables. The application was last updated in February 2018 and it is available in two versions. The trial version is accessible for free. However, it has a migration limit of 50,000 rows per table and will add an extra field in tables. The pro version costs 322$ but it has no limitations [28].

4.3 Implementation

The implementation of the requirements defined in section 4.1 is divided into two parts. In the first part, the current database running on MariaDB is extended and tested by storing MRI samples as BLOB files. In the second part, the MariaDB database is migrated to SQL Server using the tool mentioned in section 4.2.

Before proceeding to the implementation details, it is required to define BLOBs and discuss the various methods for integrating them into the database.

Binary Large Object

A Binary Large Object (BLOB) is a collection of binary data stored as a single entity in a database management system. BLOBs are typically images, videos or audio files. A BLOB column stores actual files as binary or byte strings in the database instead of a file path reference. This has the advantage that files are always available and can be displayed correctly, even if the original location has been changed or removed [29].

The use of binary large objects promises better security since data is directly stored into the database instead of being theft from simple file copying of directories. In this way, data is completely isolated from the file system. Moreover, using BLOB facilitates dealing with portability and backup issues, which improves data integrity [29].

BLOBs are mapped differently in most of the database systems. MySQL and MariaDB support four types of BLOB data types, namely TINYBLOB, BLOB, MEDIUMBLOB and LONGBLOB. These differ only in the maximum length of the value they hold. TINYBLOB only supports up to 255 bytes, BLOB up to 64 KB, while MEDIUMBLOB can handle up to 16 MB and LONGBLOB up to 4 GB [30].

On the other hand, Microsoft SQL Server provides three different solutions for storing documents in a database or on remote storage devices [31]:

- **FILESTREAM:** FILESTREAM is used to store unstructured data in the file system. Thus, applications can leverage the streaming APIs and performance of the file system, as well as maintain transactional consistency between the unstructured data and the corresponding structured data.
- **FileTables**: The FileTable feature builds on the top of existing FILESTREAM capabilities. It enables the integration and storage of data management components and provides integrated SQL Server services, including full-text search and semantic search, over unstructured data and metadata. This method allows the storage of

files in special tables called FileTables. These can be accessed through Windows applications as if they were stored in the file system.

- **Remote Blob Store (RBS)**: Remote Blob Store is a client library that enables the storage of BLOB files in storage solutions instead of saving them directly on the server. This saves a significant amount of space and avoids wasting expensive server hardware resources. However, RBS requires SQL Server Enterprise for the main database server in which the BLOB files are stored or the supplier FILESTREAM provider if working with SQL Server Standard [32].

4.3.1 Extending the Database Schema

PHP scripting language and Xampp[2] are used to interact with the MariaDB database for schema extension as well as scalability and performance testing. The connection is established using the PHP Data Objects (PDO) extension. This defines a lightweight and consistent interface for accessing databases in PHP. One of its best benefits is that it has an exception class to handle any problems that may occur in database queries. Moreover, PDO enables uncomplicated switching between different databases and platforms by simply changing the connection string [34].

The connection is set up in an external PHP-file that is invoked while interacting with the database. This is not important for the implementation, and so will not be discussed in this section (see Listing A.1).

In order to assign medical images to patients in the database, a new Table `imaging` was created as shown in Listing 4.1.

[2]Xampp (an acronym for cross (X)-platform, Apache, MySQL, PHP, and Perl) is a free and open source cross-platform, that enables running web applications locally for testing and deployment purposes [33].

```sql
CREATE OR REPLACE TABLE imaging (
 id INT(10) UNSIGNED AUTO_INCREMENT,
 user_id INT(10) UNSIGNED NOT NULL,
 filename VARCHAR(255) NOT NULL,
 type VARCHAR(255) NOT NULL,
 file MEDIUMBLOB NOT NULL,
 versionNr TINYINT NOT NULL,
 PRIMARY KEY (id),
 FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Listing 4.1: Create a new table Imaging to store files.

`Imaging` is designed to assign files of zip or dcm format to the corresponding user. The table is identified by an auto-incremented *id* and is related to the table `users` through the user id, which is stored as *user_id* in the new table in order to differentiate between the primary and foreign key. The inserted file should be given a name in the field *filename* and categorized in *type*. The type is currently defined as a textual field, yet it may be changed to an Enum in the future if the types are specified. A patient (or user) may make various scans over time. Therefore, a version is associated to scans in order to differentiate between them and is stored as a TINYINT in the column *versionNr*. Then, the actual file is stored as a MEDIUMBLOB in the column *file*. Given that MRI scans are usually bigger than 10 MB, it is considered to choose MEDIUMBLOB as a data type for the storage, which handles up to 16 MB.

Before filling the new table in the TYT database, it is required to adapt the configuration file of MySQL (my.cnf), otherwise, the system will not be able to handle such large values [30]. Consequently, it is necessary to change the message buffer size by changing the value of the `max_allowed_packet` variable, which is set per default on 1 MB, to 20 MB.

The storage of the MRI scans is realized using PHP scripting language and SQL queries. Listing 4.2 represents a script written in PHP that enables the storage of data as a BLOB type into the database. The code also aims to test the performance of the current database hosting system which will be discussed later in chapter 5.

```php
<?php
include "connect.php";
/**
* Insert BLOB file into the imaging table
*/
function insertBlob($pdo, $id, $filename, $type, $path, $versionNr) {
        $file = fopen($path, 'rb');
        $sql = 'INSERT INTO imaging
                VALUES (null,:id,:filename,:type,:file,:versionNr)';
        $stmt = $pdo->prepare($sql);
        $stmt->bindParam(':id', $id);
        $stmt->bindParam(':filename', $filename);
        $stmt->bindParam(':type', $type);
        $stmt->bindParam(':file', $file, PDO::PARAM_LOB);
        $stmt->bindParam(':versionNr', $versionNr);
        return $stmt->execute();
}
$sql ="SELECT id FROM users LIMIT 100";
$path ="samples/sample.zip";
$start = microtime(true);
foreach ($pdo->query($sql) as $row) {
        insertBlob($pdo, $row['id'],"test","MRI",$path, 1);
}
$time_elapsed = microtime(true) - $start;
echo '<b>Total Execution Time:</b> '.$time_elapsed.' secs.';
?>
```

Listing 4.2: Insert BLOB Files into the Database

PHP PDO provides a convenient way to handle BLOB data using streams and to prepare statements. To insert the content of a file into a BLOB column, it is firstly required to open the file for reading in binary using **fopen()**. This will open a file specified by its name and bind it to a stream. Then, an INSERT statement is built to enable the storage of data in the database as shown in Listing 4.3.

```
INSERT INTO imaging
VALUES (null,:id,:filename,:type,:file,:versionNr);
```

Listing 4.3: Insert Data into MariaDB Database.

All required values need to be inserted in addition to the actual file, in order to fulfil the NOT NULL constraints set by the creation of the table *imaging*. However, the id should be left empty because it is an auto-increment and so, will be automatically generated by the database. Subsequently, the prepared statement has to be bound to the file handle using the **bindParam()** method and then to be executed using the **execute()** method. *PDO::PARAM_LOB* serves to map the BLOB data as a stream.

The insertBlob function can be applied on all file types, including multimedia files, PDFs, ZIP files as well as images having the DICOM format.

For testing purposes, an MRI scan data of 13 MB will be stored to the first 100 users ids [35]. The file contains a large number of DICOM images compressed into a ZIP file. The first 100 users are selected from the table *users* and the file will be called from the file system and stored using the insertBlob function into the table *imaging*. Furthermore, it is required to determine the execution time of the procedure, in order to compare the performance of MariaDB with SQL Server later. To this end, the PHP **microtime()** method is used to measure the execution time of the loop, which will be calculated and bound to the variable *$time_elapsed*.

After the execution of the script, the data used for testing should be stored successfully in the database and the execution time will be displayed on the browser in seconds.

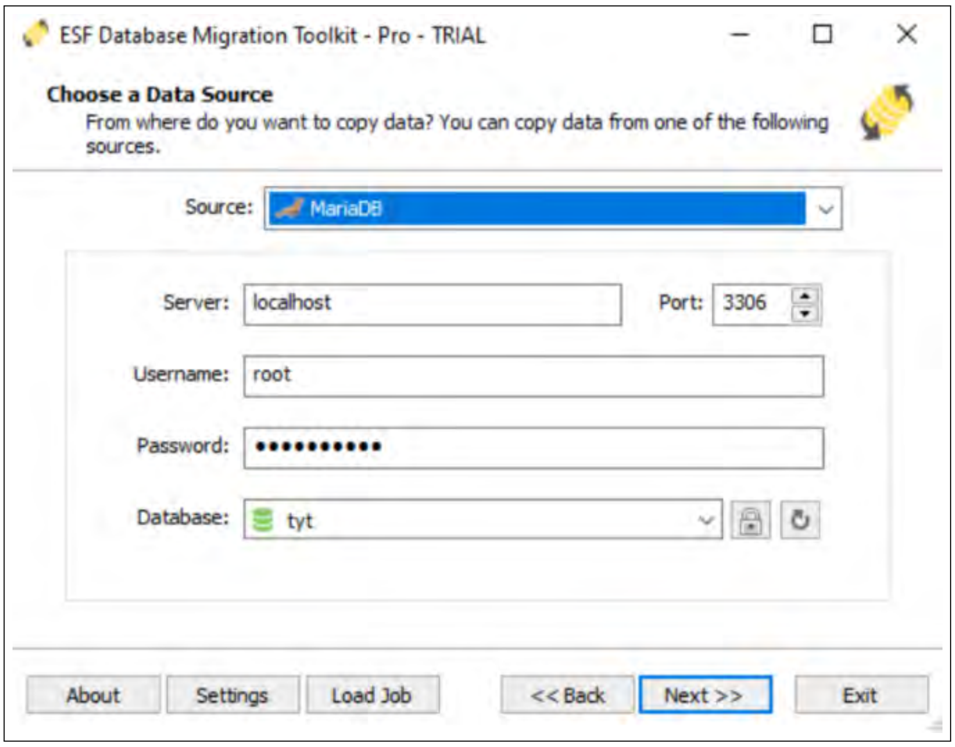
Another method was implemented to store files as BLOB data type into the new table using a graphical user interface, which enables uploading files and storing them together with the user id into the database. The script is represented in Listing A.4. The files are uploaded via a POST request, which will be contained by the PHP variable **$_FILES** and treated via the **file_get_contents()** method. The process of binding variables and storing them in the *imaging* table are similar to those described in the previous script.

4.3.2 Data Migration from MariaDB to SQL Server

Migrating very large databases from one system to another is a major challenge. It is of great relevance to consider several issues such as loss of data during the transfer, the security of the data, technical aspects of the technologies involved, etc.
In this work, it is required to transfer confidential data from MariaDB to SQL Server. To achieve this, two approaches are possible. A direct and simple way is to use a database migration tool, such as ESF Database Migration Toolkit, in order to transfer data automatically from MariaDB to SQL Server. However, the trial version of the software has some limitations (see subsection 4.2.4). The second conceivable option is to make use of a special tool provided by Microsoft SQL Server for data migration called SQL Server Migration Assistant (SSMA) [36]. The tool is designed to automate database migration to SQL Server from Microsoft Access, DB2, MySQL, Oracle and SAP ASE. Yet, MariaDB is not supported by SSMA. Consequently, an intermediate database will be required to successfully migrate data from the source to the target system. For example, the database can be migrated first from MariaDB to MySQL as an intermediate hosting system, then from MySQL to SQL Server.

Migration using ESF Database Migration Toolkit

ESF Database Migration Toolkit is chosen as a solution to accomplish the database migration from MariaDB to SQL Server. The software enables a non-complex and rapid migration process thanks to its simple graphical user interface.


ESF Database Migration Toolkit - Pro - TRIAL

Choose a Data Source

From where do you want to copy data? You can copy data from one of the following sources.

Source: MariaDB

Server: localhost

Port: 3306

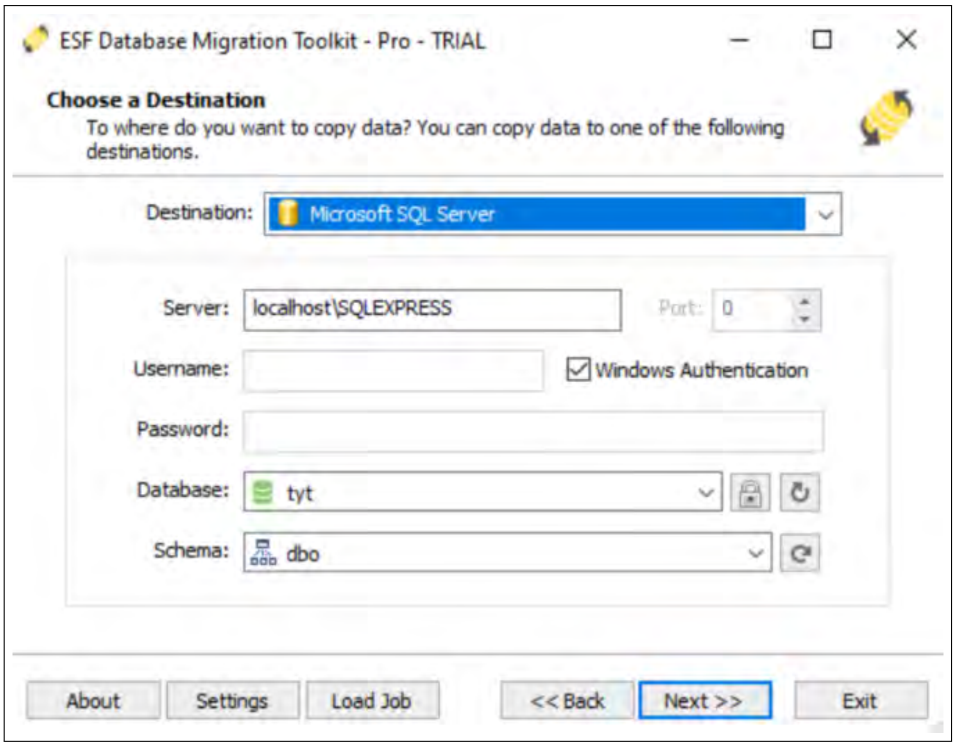
Username: root

Password: ••••••••••••

Database: tyt

Figure 4.1: Choose Source Database

Choose a Database Source First of all, it is required to connect the tool to the database source, here MariaDB. Then, the server name and the port number need to be entered to establish the connection. The TrackYourTinnitus database is running on the localhost and on the default port (port:3306). By clicking the "Refresh" button, the TYT database should appear on the list and can be selected for migration as shown in Figure 4.1.



Choose a Destination

To where do you want to copy data? You can copy data to one of the following destinations.

Destination: Microsoft SQL Server

Server: localhost\SQLEXPRESS

Port: 0

Username:

☑ Windows Authentication

Password:

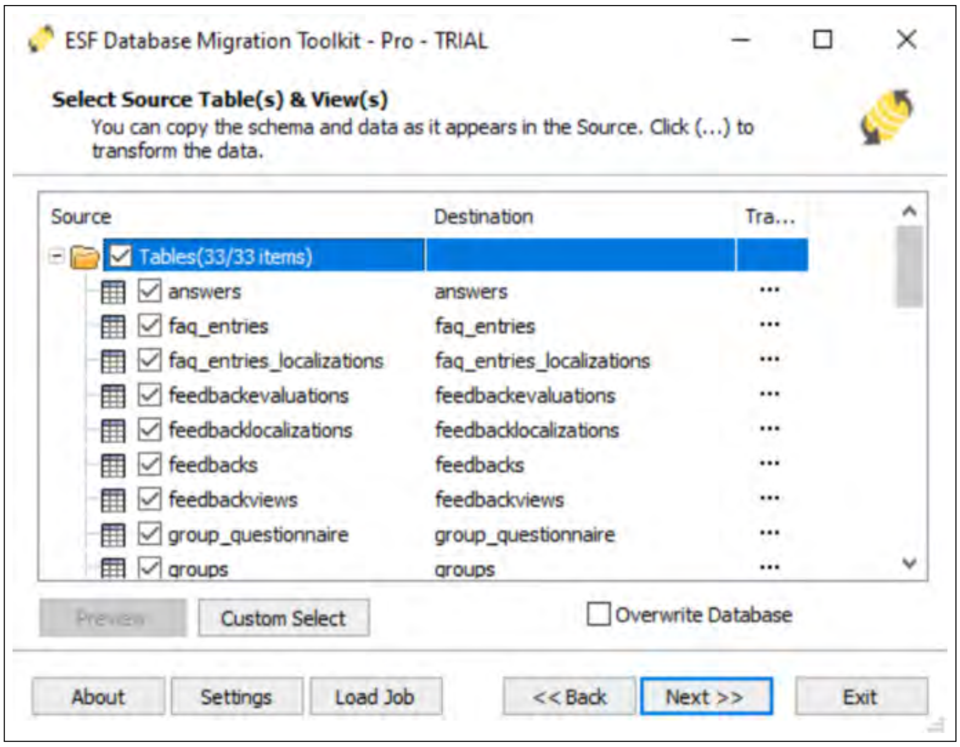
Database: tyt

Schema: dbo

Figure 4.2: Choose a Target System

Choose a Destination In the next step, Microsoft SQL Server has to be selected as a destination hosting system as shown in Figure 4.2. Then, all the information required for user identification should be entered, or it is recommended to check the option "Windows Authentication" if using Windows operating system. The target server should be named in both options, whether by inserting the SQL Server hostname or by adding an instance name, for example "localhost_SQLEXPRESS". Once the connection to the target database is set up, existing schemas will be listed and a schema can be selected for data hosting. It is also possible to enter a new schema name which will be automatically created during the migration process [37].

Select Source Tables In the next step, the existing database structure will be displayed. The tool allows manual or automatic selection of tables and views as well as renaming tables or fields before the migration process (Figure 4.3). As mentioned before, views


ESF Database Migration Toolkit - Pro - TRIAL

Select Source Table(s) & View(s)

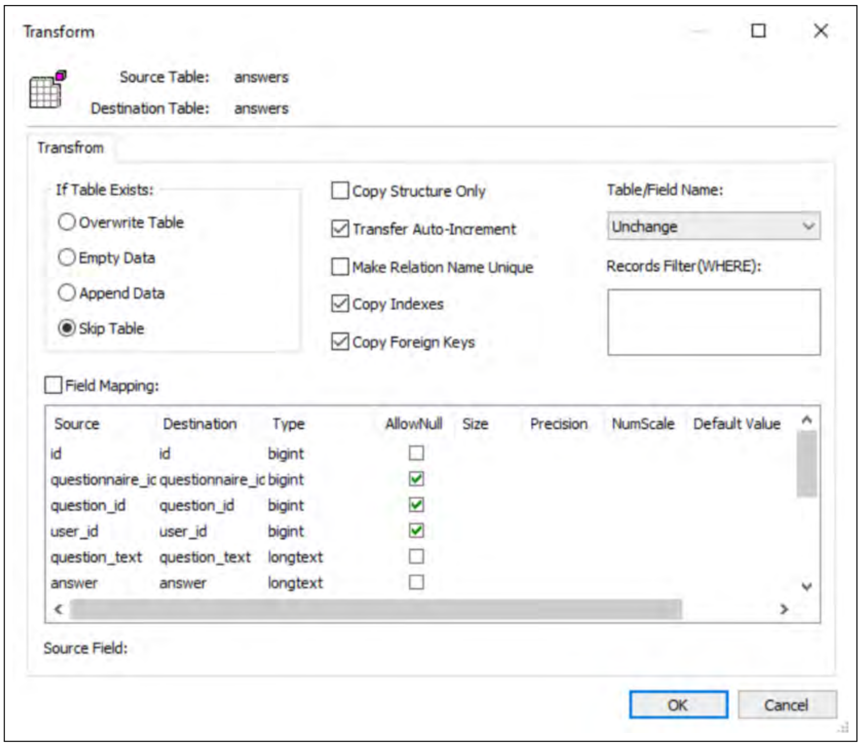
You can copy the schema and data as it appears in the Source. Click (...) to transform the data.

Source	Destination	Tra...
☑ Tables(33/33 items)		
☑ answers	answers	...
☑ faq_entries	faq_entries	...
☑ faq_entries_localizations	faq_entries_localizations	...
☑ feedbackevaluations	feedbackevaluations	...
☑ feedbacklocalizations	feedbacklocalizations	...
☑ feedbacks	feedbacks	...
☑ feedbackviews	feedbackviews	...
☑ group_questionnaire	group_questionnaire	...
☑ groups	groups	...

Figure 4.3: Select Tables for Migration

will be converted to tables. Since the TrackYourTinnitus database contains only tables and simple data objects, this will not affect the resulting system. By accessing the tables settings, it is possible to remap the table structure and filter the data before transferring as shown in Figure 4.4.

Migration Procedure After specifying the necessary information, the migration process can be started by clicking on *Submit*. The procedure will be executed within a short time and without intervention. During the conversion, the application visualizes the transfer progress and allows browsing the generated log file. When the migration process has finished, the user is informed about the migration time and status as shown in Figure 4.5. It is possible to access the full migration log via the option *Browse Log*. Additionally, the migration settings can be stored in a job file by clicking on *Save as job*. The file can then be quickly reloaded anytime.



Transform

Source Table: answers
Destination Table: answers

Transfrom

If Table Exists:
- ( ) Overwrite Table
- ( ) Empty Data
- ( ) Append Data
- (●) Skip Table

- ☐ Copy Structure Only
- ☑ Transfer Auto-Increment
- ☐ Make Relation Name Unique
- ☑ Copy Indexes
- ☑ Copy Foreign Keys

Table/Field Name: Unchange

Records Filter(WHERE):

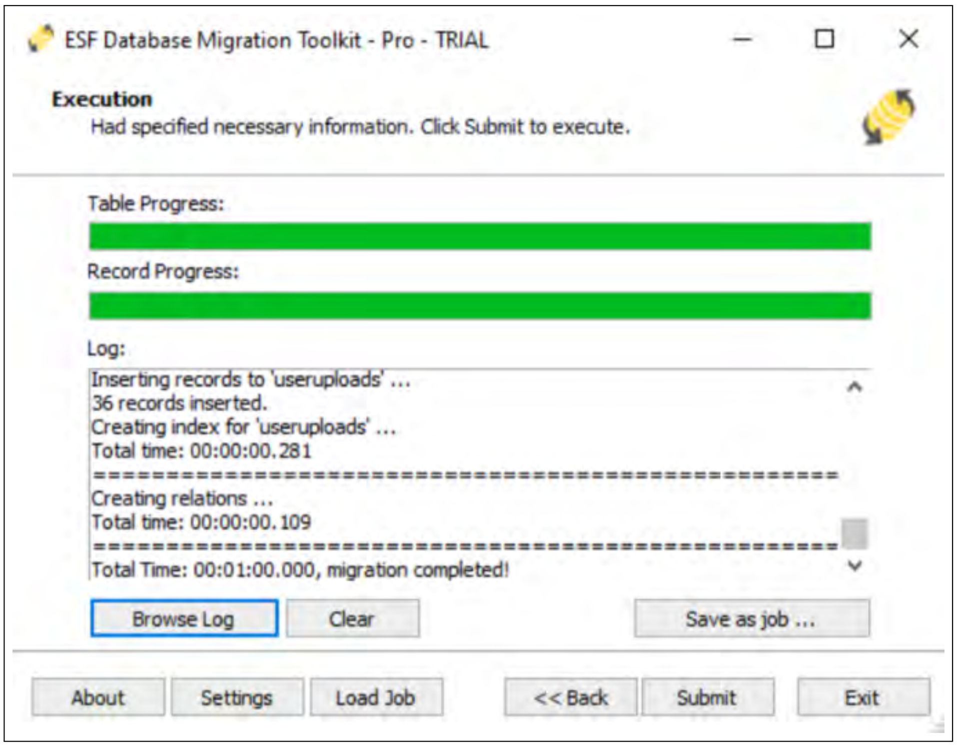
☐ Field Mapping:

Source	Destination	Type	AllowNull	Size	Precision	NumScale	Default Value
id	id	bigint	☐				
questionnaire_ic	questionnaire_ic	bigint	☑				
question_id	question_id	bigint	☑				
user_id	user_id	bigint	☑				
question_text	question_text	longtext	☐				
answer	answer	longtext	☐				

Source Field:

Figure 4.4: Advanced Options

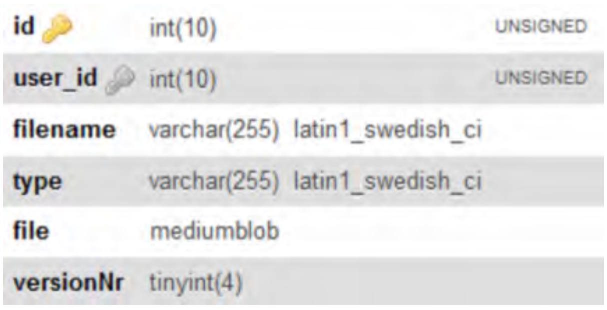
The migration process was completed successfully and all selected tables were transferred to SQL Server. A detailed course of the migration is demonstrated in the log file, which is appended in the annex.



```
Inserting records to 'useruploads' ...
36 records inserted.
Creating index for 'useruploads' ...
Total time: 00:00:00.281
====================================================
Creating relations ...
Total time: 00:00:00.109
====================================================
Total Time: 00:01:00.000, migration completed!
```

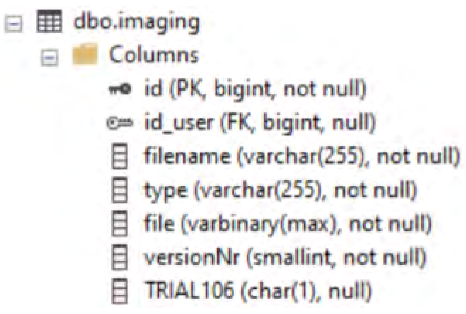
Figure 4.5: Executing Database Migration

The TYT database hosted in SQL Server can be accessed via SSMS. The application offers a well-structured overview of tables and columns and enables the execution of queries using T-SQL. The configurations set over the migration tool were adopted and the data types were automatically converted into the T-SQL syntax, including the BLOB data type. Figure 4.6 and Figure 4.7 represent the data structure of the table *imaging* in MariaDB and SQL Server. It is notable that T-SQL has many similarities with the standard language, but some data types, methods and statements are defined differently in comparison with SQL. For example, **int** in SQL will be declared as **bigint** in T-SQL and **tinyint** as **smallint**. Besides, MariaDB offers four different data types to store BLOBs while those are defined as **varbinary(max)** in SQL Server, regardless of their size. The storage of BLOB files is also implemented differently in both databases. As shown in Listing 4.2, a script written in PHP was used in order to connect to the MariaDB database and fill the *imaging* table with MRI samples, unlike SQL Server which enables the direct storage of data using T-SQL. The Insert-Statement is run on SSMS and is constructed



id	int(10)		UNSIGNED
user_id	int(10)		UNSIGNED
filename	varchar(255)	latin1_swedish_ci	
type	varchar(255)	latin1_swedish_ci	
file	mediumblob		
versionNr	tinyint(4)		

Figure 4.6: Imaging in MariaDB



```
dbo.imaging
  Columns
    id (PK, bigint, not null)
    id_user (FK, bigint, null)
    filename (varchar(255), not null)
    type (varchar(255), not null)
    file (varbinary(max), not null)
    versionNr (smallint, not null)
    TRIAL106 (char(1), null)
```

Figure 4.7: Imaging in SQL Server

as below: First, the records which need to be stored are selected. Then, they will be inserted into the adequate table. The BLOB file is imported via **Openrowset**, which returns the content of the binary data as a single rowset varbinay(max) output. Moreover, an alias should be assigned to the BLOB column, otherwise, the query will return an error. Listing 4.4 represents an example of how an INSERT-Statement that deals with a blob data type looks like.

```sql
INSERT imaging (id,id_user,filename,type,versionNr,file)
SELECT 1,1,'BLOB File', 'MRI', 1, BulkColumn FROM Openrowset
( Bulk 'C:\samples\sample.zip', Single_Blob) AS tb
```

Listing 4.4: Insert BLOB Data in SQL Server using T-SQL

5 Evaluation

This chapter contains an interpretation of the results obtained from running the methods described throughout this work on the current relational database hosting system as well as the defined database management system for the data migration. It will also feature arguments for why the problems tackled and solutions described in this project might be interesting in the future.

5.1 Results

First, the migration process and the generated database in SQL Server are evaluated. Then, the performance of both database hosting systems is compared by running the same queries. The results will reveal the strength and the downsides of MariaDB and SQL Server and whether the migration to SQL Server is recommendable or not.

5.1.1 Database Migration

ESF Database Migration Toolkit has proven to be a very helpful tool for transferring database contents. The tool does not only support a wide range of databases, but also allows an easy migration process within a short time. As shown in Figure 4.5, the transfer of over 200.000 datasets across 33 tables was done within one minute.
After the migration, the database was stored in SQL Server and could be accessed via SSMS. Moreover, the configurations set through the tool were applied and the data types were automatically converted to the SQL Server, including the BLOB data type. Nevertheless, the trial version of ESF Database Migration Toolkit has its disadvantages.

The tool offers the possibility to transfer an infinite number of tables. However, it sets a migration limit of 50.000 datasets per table and the remaining data will be lost. From the 33 tables that belong to the TrackYourtinnitus database, two tables contain more than the number mentioned before: *answers* and *standardanswers* (see Figure 5.1). As a result, personal as well as general information of users were transferred properly in the generated database, but some of the answers gathered from filling questionnaires were lost. Another limitation of the trial version is that extra columns are added to tables. Indeed, an empty column having the name *TRIAL_xxx* and the constraint NULL was generated at the bottom of each table, e.g. TRIAL106 in the table imaging represented in Figure 4.7.

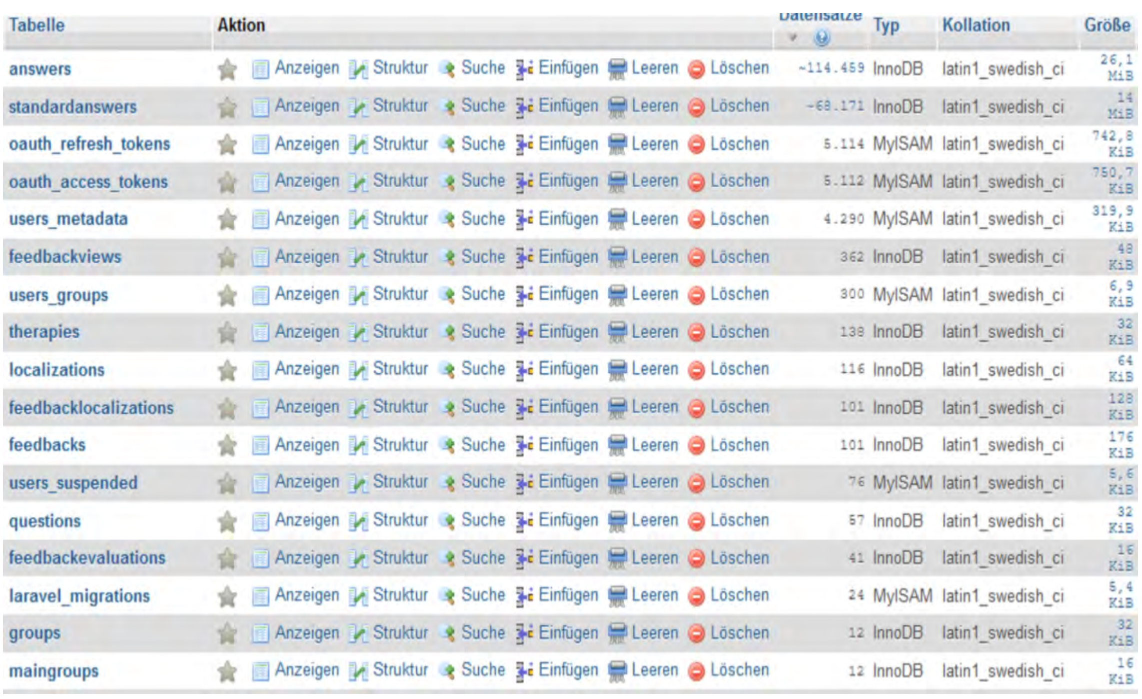


Tabelle	Aktion	Datensätze	Typ	Kollation	Größe
answers	Anzeigen Struktur Suche Einfügen Leeren Löschen	~114.459	InnoDB	latin1_swedish_ci	26,1 MiB
standardanswers	Anzeigen Struktur Suche Einfügen Leeren Löschen	~68.171	InnoDB	latin1_swedish_ci	14 MiB
oauth_refresh_tokens	Anzeigen Struktur Suche Einfügen Leeren Löschen	5.114	MyISAM	latin1_swedish_ci	742,8 KiB
oauth_access_tokens	Anzeigen Struktur Suche Einfügen Leeren Löschen	5.112	MyISAM	latin1_swedish_ci	750,7 KiB
users_metadata	Anzeigen Struktur Suche Einfügen Leeren Löschen	4.290	MyISAM	latin1_swedish_ci	319,9 KiB
feedbackviews	Anzeigen Struktur Suche Einfügen Leeren Löschen	362	InnoDB	latin1_swedish_ci	48 KiB
users_groups	Anzeigen Struktur Suche Einfügen Leeren Löschen	300	MyISAM	latin1_swedish_ci	6,9 KiB
therapies	Anzeigen Struktur Suche Einfügen Leeren Löschen	138	InnoDB	latin1_swedish_ci	32 KiB
localizations	Anzeigen Struktur Suche Einfügen Leeren Löschen	116	InnoDB	latin1_swedish_ci	64 KiB
feedbacklocalizations	Anzeigen Struktur Suche Einfügen Leeren Löschen	101	InnoDB	latin1_swedish_ci	128 KiB
feedbacks	Anzeigen Struktur Suche Einfügen Leeren Löschen	101	InnoDB	latin1_swedish_ci	176 KiB
users_suspended	Anzeigen Struktur Suche Einfügen Leeren Löschen	76	MyISAM	latin1_swedish_ci	5,6 KiB
questions	Anzeigen Struktur Suche Einfügen Leeren Löschen	57	InnoDB	latin1_swedish_ci	32 KiB
feedbackevaluations	Anzeigen Struktur Suche Einfügen Leeren Löschen	41	InnoDB	latin1_swedish_ci	16 KiB
laravel_migrations	Anzeigen Struktur Suche Einfügen Leeren Löschen	24	MyISAM	latin1_swedish_ci	5,4 KiB
groups	Anzeigen Struktur Suche Einfügen Leeren Löschen	12	InnoDB	latin1_swedish_ci	32 KiB
maingroups	Anzeigen Struktur Suche Einfügen Leeren Löschen	12	InnoDB	latin1_swedish_ci	16 KiB

Figure 5.1: TYT Tables ordered by their Size

5.1.2 MariaDB vs SQL Server Performance

MariaDB and Microsoft SQL Server are two of the most popular relational database management systems. The choice of the database platform is very important because this will end up being the core of the dynamic content moving forward. The decision hinges on many factors such as the performance, speed as well as security.

In this part, the performance of MariaDB and SQL Server is analysed in order to discover which system is well suited for the TrackYourTinnitus platform. Different SELECT, INSERT and DELETE queries were executed on both databases on the Windows system and their execution time was recorded. The results are shown in Table 5.1.

Query	**MariaDB**	**SQL SERVER**
INSERT-Statement	42 secs	12 secs
UPDATE imaging SET filename = 'MRI scan'	10 secs	0 secs
SELECT * FROM imaging	3 secs	15 secs
DELETE FROM imaging	9 secs	0 sec
SELECT * FROM answers	**Fatal error:** Maximum execution time of 300 seconds exceeded	1 sec

Table 5.1: Comparison of the execution time in MariaDB and SQL Server

The INSERT query was enclosed in a loop to store zip files for a large number of users in the table *imaging*. On the first try, the MariaDB system crashed down while trying to store files having a size of 13 MB for each user. Therefore, the loop was limited to 100 records as shown in Listing 4.2. The script was run several times and the execution time was always between 41 and 47 seconds, which is considered to be very slow. The same test was done on SQL Server using the query mentioned in Listing 4.1 for 100 users and the query took on average 12 seconds to complete. The execution time shows that MariaDB took almost four times of the time taken by SQL Server for the data insertion. However, the most significant difference between the two was seen in terms of SELECT statements of a large number of data. For 50000 rows SELECT statement, MariaDB ejected an error because the execution time exceeded 300 seconds, while SQL Server took only one second to display all records.

5.2 Discussion

Returning to the question posed at the beginning of this thesis, it is now reasonable to state that SQL Server is a suitable replacement for MariaDB for hosting the TrackYourTinnitus database. The queries used for analytical purposes showed that SQL Server offers better performance than MariaDB in terms of response time. Curious is that selecting the records, in which the files were contained, took notably less time in MariaDB in comparison with SQL Server. Thus, SQL Server consistently took lesser time for all the other test cases compared with MariaDB. Additionally, SQL Server comes with many useful tools that facilitate not only the interaction with databases and running queries on a graphical user interface but also database migration from several relational database management systems to SQL Server using Integration Services. This has not been tested given that it does not support MariaDB, so it remains an open question whether database migration using SSIS would be lossy or not. Instead, ESF Database Migration Toolkit was used to perform the migration. It can be said that the data transfer was successful, although some user answers were lost. However, migrating databases has always been very challenging without any data loss. To solve this problem, the missing data can be converted manually, for example by writing a script that splits the tables that exceed the row limit number or by extending the data to an extra table in Maria DB. Then, the migration process can be realised as usual. Afterwards, the data shall be merged with the existing data. Another issue was the addition of an empty column to each table which raised the size of the TYT database from 24 MB to 144 MB. Since the TYT contains only 33 tables, it is feasible to delete the undesirable columns manually using SSMS.

In a nutshell, switching to SQL Server is beneficial for the TrackYourTinnitus platform. MSSQL does not only offer better security and includes advanced features, but it also offers better performance compared to MariaDB. Yet, it is necessary to test the new system on the back-end to avoid application failure.

6 Conclusion

In this chapter, the results of this paper in terms of the extensions implemented and the tools used are summarized again. Then, the chapter ends with an outlook on possible improvements of the application as well as the TrackYourTinnitus database that might be interesting for the future if this work is carried on.

6.1 Summary

In this thesis, Microsoft SQL Server has been considered as an alternative system to host the TrackYourTinnitus database, which is currently running on MariaDB. Many factors are involved in this decision. The main reason behind this switchover is that the data of the TrackYourTinnitus will be merged with clinical databases and then MRI images will be added to its, which means that the amount of data in the TYT database will hugely increase drastically in the future. MariaDB is one of the most widely used relational databases. Yet, when it comes to hosting a large amount of data, it is not the best option on the market. Furthermore, the Framework barrier is a hurdle while choosing an alternative to MariaDB, since the Laravel framework, which is used in the implementation of the back-end, only supports few database systems. As a consequence, SQL Server has been chosen due to its compatibility with Laravel and for its advanced functions offered by T-SQL. The RDBMS is commercial and available in different editions. The express edition has been used to fulfil the requirements mentioned before, since it is free and can hold databases of a size up to 10 GB, which is well enough for hosting the TYT database. Then, the data was transferred from MariaDB to SQL Server using ESF Database Migration Toolkit. The tool supports a wide range of databases and enables an easy and rapid system migration. Afterwards, MRI files were uploaded and stored as a BLOB column on both databases. MariaDB offers different data types for storage such as MEDIUMBLOB, which handles up to 16 MB, while SQL Server allows the storage of binary files as varbinary data type. In addition to that, SQL Server provides more solutions for storing documents such as FILESTREAM, FileTables and Remote Blob Store. MariaDB does not include similar tools. Then, the performance of both databases was compared by running the same queries and recording the execution time taken by each query. The results have revealed that SQL Server offers better performance than MariaDB in terms of response time.

All things considered, it seems reasonable to assume that SQL Server is a suitable replacement for MariaDB for hosting the TrackYourTinnitus database. MSSQL provides high scalability and ensures smooth database operations when dealing with data on a large scale.

6.2 Future Work

Both database management systems used within this thesis were tested based on the execution time of different SELECT, INSERT and DELETE queries. Thus, the results may differ if tested on the TYT application. Hence, it is recommendable to integrate the database hosted in SQL Server in the back-end of the TYT platform in order to test the compatibility as well as the performance of the system. Moreover, the database had to be expanded with BLOB columns, which serve to assign MRI images to patients. For this purpose, a client-side application should be implemented to deal with these new features. Since most medical data is present in the DICOM format, adding a DICOM viewer to the TYT application should be considered so that MRI scans can be displayed on the web browser.

Bibliography

[1] Mayo Foundation for Medical Education and Research (MFMER): Tinnitus - Symptoms and causes . https://www.mayoclinic.org/diseases-conditions/tinnitus/symptoms-causes/syc-20350156 (2019)

[2] Pryss, R., Probst, T., Schlee, W., Schobel, J., Langguth, B., Neff, P., Spiliopoulou, M., Reichert, M.: Prospective crowdsensing versus retrospective ratings of tinnitus variability and tinnitus–stress associations based on the TrackYourTinnitus mobile platform. International Journal of Data Science and Analytics (2018)

[3] Pryss, R., Reichert, M., Langguth, B., Schlee, W.: Mobile Crowd Sensing Services for Tinnitus Assessment, Therapy, and Research. Proceedings - 2015 IEEE 3rd International Conference on Mobile Services, MS 2015 (2015)

[4] Laravel: Database: Getting started. https://laravel.com/docs/6.x/database (2019)

[5] Herrmann, J.: Konzeption und technische Realisierung eines mobilen Frameworks zur Unterstützung tinnitusgeschädigter Patienten. (2014)

[6] Motwani, R., Reichert, M., Kalle, S., Pryss, R., Schlee, W., Probst, T., Langguth, B., Landgrebe, M., Spiliopoulou, M.: Studying the potential of multi-target classification on patient screening data to predict dropout cases. In: 2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS). (2018)

[7] Probst, T., Pryss, R.C., Langguth, B., Spiliopoulou, M., Landgrebe, M., Vesala, M., Harrison, S., Schobel, J., Reichert, M., Stach, M., Schlee, W.: Outpatient Tinnitus Clinic, Self-Help Web Platform, or Mobile Application to Recruit Tinnitus Study Samples? Frontiers in Aging Neuroscience (2017)

[8] Hearing Link: What is tinnitus? https://www.hearinglink.org/your-hearing/tinnitus/what-is-tinnitus/ (2018)

[9] Knör, E.: Die Deutsche Tinnitus-Liga e.V. (DTL). (HNO Praxis heute Tinnitus) 173–180

[10] Liu, J., Shen, H., Narman, H.S., Chung, W., Lin, Z.: A Survey of Mobile Crowdsensing Techniques: A Critical Component for The Internet of Things. ACM Trans. Cyber-Phys. Syst. (2018)

[11] Wikipedia contributors: Crowdsensing — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Crowdsensing&oldid=928716304 (2019) [Online; accessed 25-December-2019].

[12] Pryss, R., Schlee, W., Langguth, B., Reichert, M.: Mobile Crowdsensing Services for Tinnitus Assessment and Patient Feedback. Proceedings - 2017 IEEE 6th International Conference on AI and Mobile Services, AIMS 2017 (2017)

[13] Wikipedia contributors: Medical imaging — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Medical_imaging&oldid=931073622 (2019) [Online; accessed 23-December-2019].

[14] NEMA: DICOM Standard. https://www.dicomstandard.org/ (2019)

[15] : DICOM Bild und Daten. https://www.dateiendung.com/format/dicom (2019)

[16] National Institute of Biomedical Imaging and Bioengineering (NIBIB): Magnetic Resonance Imaging (MRI). https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri (2019)

[17] Fielding, R.T.: Representational State Transfer (REST). https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (2000)

[18] Wikipedia contributors: PHP — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=PHP&oldid=928911210 (2019) [Online; accessed 3-December-2019].

[19] Wikipedia contributors: Laravel — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Laravel&oldid=931730483 (2019) [Online; accessed 15-December-2019].

[20] Rouse, M.: Microsoft SQL Server. https://searchsqlserver.techtarget.com/definition/SQL-Server (2019)

[21] DB-Engines: System Properties Comparison MariaDB vs. Microsoft SQL Server vs. MySQL. https://db-engines.com/en/system/MariaDB%3BMicrosoft+SQL+Server%3BMySQL (2019)

[22] Potter, J.: MySQL Performance: MySQL vs. MariaDB. https://www.liquidweb.com/kb/mysql-performance-mysql-vs-mariadb/ (2018)

[23] MariaDB Foundation: About MariaDB Server. https://mariadb.org/about/ (2019)

[24] Atlantic.Net: What is MSSQL? https://www.atlantic.net/what-is-mssql/ (2019)

[25] Mistry, R., Misner, S.: Introducing Microsoft SQL Server 2019. (2019)

[26] Rouse, M., Ferguson, K., Viescas, J., Sheldon, R.: What is T-SQL (Transact-SQL)? https://searchsqlserver.techtarget.com/definition/T-SQL (2019)

[27] Chugugrace: SQL Server Integration Services - SQL Server Integration Services (SSIS). https://docs.microsoft.com/en-us/sql/integration-services/sql-server-integration-services?view=sql-server-ver15 (2018)

[28] Zhang, P.: Practical Guide to Large Database Migration. CRC Press (2019)

[29] Ndungu, F.: How to Work With BLOB in a MySQL Database Hosted on Alibaba Cloud. https://dzone.com/articles/how-to-work-with-blob-in-mysql-database-hosted-on (2019)

[30] MySQL Documentation: The BLOB and TEXT Types. (https://dev.mysql.com/doc/refman/8.0/en/blob.html)

[31] MikeRayMSFT: Binary Large Object (Blob) Data (SQL Server) . =https://docs.microsoft.com/en-us/sql/relational-databases/blob/binary-large-object-blob-data-sql-server?view=sql-server-ver15 (2017)

[32] SQL Server: Remote Blob Store (RBS) (SQL Server). https://docs.microsoft.com/en-us/sql/relational-databases/blob/remote-blob-store-rbs-sql-server?view=sql-server-ver15 (2016)

[33] Yusuf, R.: What is the use of XAMPP? https://www.quora.com/What-is-the-use-of-XAMPP (2018)

[34] JavaTPoint: Introduction to PHP PDO. (https://www.javatpoint.com/php-pdo)

[35] VIEWER, J.D.: DICOM Viewer to view medical data. (https://www.visus.com/en/downloads/jivex-dicom-viewer.html)

[36] : SQL Server Migration Assistant - SQL Server_2019. https://docs.microsoft.com/de-de/sql/ssma/sql-server-migration-assistant?view=sql-server-ver15 (2019)

[37] Wong, M.: Migrating data from MariaDB to SQL Server: DBSofts. https://www.dbsofts.com/articles/mariadb_to_sql_server/ (2019)

A
Sources

This appendix contains important source code snippets implemented during this work.

```php
<?php
/**
* Connect MariaDB Database with PHP Using PDO.
**/
function connect()
{
  $host = "localhost";
  $user = "root";
  $pw = "";
  $db = "tyt";
  try {
   $conn = new PDO("mysql:host=$host;dbname=$db", $user, $pw);
   //set the PDO error mode to exception
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  }
  catch (PDOException $e) {
   die ("Connection failed: " . $e->getMessage());
  }
  return $conn;
}
$pdo = connect();
?>
```

Listing A.1: connect.php

```php
<?php
/**
 * Create new Imaging table to add documents to the appropriate users
 * Source: https://www.w3schools.com/php/php_mysql_create_table.asp
 */
include "connect.php";

#sql to create table
$sql = "CREATE OR REPLACE TABLE imaging (
    id INT(10) UNSIGNED AUTO_INCREMENT,
    user_id INT(10) UNSIGNED NOT NULL,
    filename VARCHAR(255) NOT NULL,
    type VARCHAR(255) NOT NULL, #Type of the file
    file MEDIUMBLOB NOT NULL,
    versionNr TINYINT NOT NULL, #Version of the file
    PRIMARY KEY (id),
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
    )";

#use exec() because no results are returned
$pdo->exec($sql);
echo "Table Imaging created successfully";
?>
```

Listing A.2: Create a new table Imaging in the database

```php
<?php
/**
 * This script tests the performance of the DB by
 * storing an MRI scan for the first 100 patients.
 * Source: https://www.mysqltutorial.org/php-mysql-blob/
 */
include "connect.php";
/**
 * Insert blob into the imaging table
 */
function insertBlob($pdo, $id, $filename, $type, $path, $versionNr) {
    $file = fopen($path, 'rb');
    $sql = 'INSERT INTO imaging
            VALUES(null,:id,:filename,:type,:file,:versionNr)';
    $stmt = $pdo->prepare($sql);
    $stmt->bindParam(':id', $id);
    $stmt->bindParam(':filename', $filename);
    $stmt->bindParam(':type', $type);
    $stmt->bindParam(':file', $file, PDO::PARAM_LOB);
    $stmt->bindParam(':versionNr', $versionNr);
    return $stmt->execute();
}

$sql ="SELECT id FROM users LIMIT 100";
$path ="samples/sample.zip";
$start = microtime(true);

foreach ($pdo->query($sql) as $row) {
    insertBlob($pdo, $row['id'],"test","MRI",$path, rand(1,100));
}
$time_elapsed = microtime(true) - $start;
echo '<b>Total Execution Time:</b> '.$time_elapsed.' secs.';
?>
```

Listing A.3: Test performance by storing BLOB files for 100 Users

```php
<?php
/**
 * This script serves to upload MRI scans and
 * store them as a blob file into the database
 */
include 'connect.php';

if (isset($_POST['submit'])) {
    $id = $_POST['id'];
    $filename = $_POST['filename'];
    $type = $_POST['type'];
    $file = file_get_contents($_FILES['blob']['tmp_name']);
    $versionNr = $_POST['versionNr'];

    $stmt = $pdo->prepare('INSERT INTO imaging VALUES
                         (null,:id,:filename,:type,:file,:versionNr)');
    $stmt->bindParam(':id', $id);
    $stmt->bindParam(':filename', $filename);
    $stmt->bindParam(':type', $type);
    $stmt->bindParam(':file', $file);
    $stmt->bindParam(':versionNr', $versionNr);
    try {
        $stmt->execute();
    }
    catch (PDOException $e) {
        echo 'Connection failed: ' . $e->getMessage();
    }
}
?>
```

Listing A.4: Store MRI Files Using the POST Request

Migration Log File

```
ESF Database Migration Toolkit - Pro - TRIAL 10.0.25

http://www.dbsofts.com. All rights reserved.

=====================================================

Connecting SOURCE('tyt') 'MariaDB' ...

Connecting DEST('tyt') 'Microsoft SQL Server' ...

=====================================================

Getting 'answers' table structure ...

Creating table 'answers' ...

Total records: 115499

Inserting records to 'answers' ...

Estimated time remaining: 00:00:20, Position: 5000/50000

Estimated time remaining: 00:00:16, Position: 10000/50000

Estimated time remaining: 00:00:14, Position: 15000/50000

Estimated time remaining: 00:00:11, Position: 20000/50000

Estimated time remaining: 00:00:09, Position: 25000/50000

Estimated time remaining: 00:00:07, Position: 30000/50000

Estimated time remaining: 00:00:05, Position: 35000/50000

Estimated time remaining: 00:00:03, Position: 40000/50000

Estimated time remaining: 00:00:01, Position: 45000/50000

Estimated time remaining: 00:00:00, Position: 50000/50000

50000 records inserted.

Creating index for 'answers' ...

Total time: 00:00:19.047

=====================================================

Getting 'faq_entries' table structure ...

Creating table 'faq_entries' ...

Total records: 2

Inserting records to 'faq_entries' ...

2 records inserted.
```

```
Creating index for 'faq_entries' ...
Total time: 00:00:00.234
====================================================
Getting 'faq_entries_localizations' table structure ...
Creating table 'faq_entries_localizations' ...
Total records: 2
Inserting records to 'faq_entries_localizations' ...
2 records inserted.
Creating index for 'faq_entries_localizations' ...
Total time: 00:00:00.219
====================================================
Getting 'feedbackevaluations' table structure ...
Creating table 'feedbackevaluations' ...
Total records: 41
Inserting records to 'feedbackevaluations' ...
41 records inserted.
Creating index for 'feedbackevaluations' ...
Total time: 00:00:00.203
====================================================
Getting 'feedbacklocalizations' table structure ...
Creating table 'feedbacklocalizations' ...
Total records: 101
Inserting records to 'feedbacklocalizations' ...
101 records inserted.
Creating index for 'feedbacklocalizations' ...
Total time: 00:00:00.250
====================================================
Getting 'feedbacks' table structure ...
Creating table 'feedbacks' ...
Total records: 101
Inserting records to 'feedbacks' ...
```

```
101 records inserted.
Creating index for 'feedbacks' ...
Total time: 00:00:00.703
====================================================
Getting 'feedbackviews' table structure ...
Creating table 'feedbackviews' ...
Total records: 362
Inserting records to 'feedbackviews' ...
362 records inserted.
Creating index for 'feedbackviews' ...
Total time: 00:00:00.359
====================================================
Getting 'group_questionnaire' table structure ...
Creating table 'group_questionnaire' ...
Total records: 0
0 records inserted.
Creating index for 'group_questionnaire' ...
Total time: 00:00:00.188
====================================================
Getting 'groups' table structure ...
Creating table 'groups' ...
Total records: 12
Inserting records to 'groups' ...
12 records inserted.
Creating index for 'groups' ...
Total time: 00:00:00.344
====================================================
Getting 'imaging' table structure ...
Creating table 'imaging' ...
Total records: 0
0 records inserted.
```

```
Creating index for 'imaging' ...
Total time: 00:00:00.188
====================================================
Getting 'laravel_migrations' table structure ...
Creating table 'laravel_migrations' ...
Total records: 24
Inserting records to 'laravel_migrations' ...
24 records inserted.
Creating index for 'laravel_migrations' ...
Total time: 00:00:00.187
====================================================
Getting 'localizations' table structure ...
Creating table 'localizations' ...
Total records: 116
Inserting records to 'localizations' ...
116 records inserted.
Creating index for 'localizations' ...
Total time: 00:00:00.281
====================================================
Getting 'maingroups' table structure ...
Creating table 'maingroups' ...
Total records: 12
Inserting records to 'maingroups' ...
12 records inserted.
Creating index for 'maingroups' ...
Total time: 00:00:00.406
====================================================
Getting 'maingroupuservalues' table structure ...
Creating table 'maingroupuservalues' ...
Total records: 4
Inserting records to 'maingroupuservalues' ...
```

```
4 records inserted.
Creating index for 'maingroupuservalues' ...
Total time: 00:00:00.453
====================================================
Getting 'oauth_access_tokens' table structure ...
Creating table 'oauth_access_tokens' ...
Total records: 5112
Inserting records to 'oauth_access_tokens' ...
5112 records inserted.
Creating index for 'oauth_access_tokens' ...
Total time: 00:00:00.985
====================================================
Getting 'oauth_auth_codes' table structure ...
Creating table 'oauth_auth_codes' ...
Total records: 0
0 records inserted.
Creating index for 'oauth_auth_codes' ...
Total time: 00:00:00.203
====================================================
Getting 'oauth_clients' table structure ...
Creating table 'oauth_clients' ...
Total records: 6
Inserting records to 'oauth_clients' ...
6 records inserted.
Creating index for 'oauth_clients' ...
Total time: 00:00:00.187
====================================================
Getting 'oauth_refresh_tokens' table structure ...
Creating table 'oauth_refresh_tokens' ...
Total records: 5114
Inserting records to 'oauth_refresh_tokens' ...
```

```
5114 records inserted.
Creating index for 'oauth_refresh_tokens' ...
Total time: 00:00:01.297
====================================================
Getting 'questionnairelocalizations' table structure ...
Creating table 'questionnairelocalizations' ...
Total records: 8
Inserting records to 'questionnairelocalizations' ...
8 records inserted.
Creating index for 'questionnairelocalizations' ...
Total time: 00:00:00.375
====================================================
Getting 'questionnaires' table structure ...
Creating table 'questionnaires' ...
Total records: 4
Inserting records to 'questionnaires' ...
4 records inserted.
Creating index for 'questionnaires' ...
Total time: 00:00:00.266
====================================================
Getting 'questions' table structure ...
Creating table 'questions' ...
Total records: 57
Inserting records to 'questions' ...
57 records inserted.
Creating index for 'questions' ...
Total time: 00:00:00.296
====================================================
Getting 'rules' table structure ...
Creating table 'rules' ...
Total records: 4
```

```
Inserting records to 'rules' ...
4 records inserted.
Creating index for 'rules' ...
Total time: 00:00:00.485
=====================================================
Getting 'standardanswers' table structure ...
Creating table 'standardanswers' ...
Total records: 69521
Inserting records to 'standardanswers' ...
Estimated time remaining: 00:00:26, Position: 5000/50000
Estimated time remaining: 00:00:22, Position: 10000/50000
Estimated time remaining: 00:00:19, Position: 15000/50000
Estimated time remaining: 00:00:16, Position: 20000/50000
Estimated time remaining: 00:00:13, Position: 25000/50000
Estimated time remaining: 00:00:11, Position: 30000/50000
Estimated time remaining: 00:00:08, Position: 35000/50000
Estimated time remaining: 00:00:05, Position: 40000/50000
Estimated time remaining: 00:00:02, Position: 45000/50000
Estimated time remaining: 00:00:00, Position: 50000/50000
50000 records inserted.
Creating index for 'standardanswers' ...
Total time: 00:00:28.187
=====================================================
Getting 'therapies' table structure ...
Creating table 'therapies' ...
Total records: 138
Inserting records to 'therapies' ...
138 records inserted.
Creating index for 'therapies' ...
Total time: 00:00:00.282
=====================================================
Getting 'tipp_entries' table structure ...
Creating table 'tipp_entries' ...
Total records: 2
Inserting records to 'tipp_entries' ...
2 records inserted.
Creating index for 'tipp_entries' ...
Total time: 00:00:00.187
====================================================
Getting 'tipp_entries_localizations' table structure ...
Creating table 'tipp_entries_localizations' ...
Total records: 1
Inserting records to 'tipp_entries_localizations' ...
1 records inserted.
Creating index for 'tipp_entries_localizations' ...
Total time: 00:00:00.187
====================================================
Getting 'userfeedbackgroups' table structure ...
Creating table 'userfeedbackgroups' ...
Total records: 4
Inserting records to 'userfeedbackgroups' ...
4 records inserted.
Creating index for 'userfeedbackgroups' ...
Total time: 00:00:00.203
====================================================
Getting 'usermemories' table structure ...
Creating table 'usermemories' ...
Total records: 2
Inserting records to 'usermemories' ...
2 records inserted.
Creating index for 'usermemories' ...
Total time: 00:00:00.563
```

```
=====================================================
Getting 'users' table structure ...
Creating table 'users' ...
Total records: 4276
Inserting records to 'users' ...
4276 records inserted.
Creating index for 'users' ...
Total time: 00:00:00.890
=====================================================
Getting 'users_groups' table structure ...
Creating table 'users_groups' ...
Total records: 300
Inserting records to 'users_groups' ...
300 records inserted.
Total time: 00:00:00.297
=====================================================
Getting 'users_metadata' table structure ...
Creating table 'users_metadata' ...
Total records: 4290
Inserting records to 'users_metadata' ...
4290 records inserted.
Creating index for 'users_metadata' ...
Total time: 00:00:00.969
=====================================================
Getting 'users_suspended' table structure ...
Creating table 'users_suspended' ...
Total records: 76
Inserting records to 'users_suspended' ...
76 records inserted.
Creating index for 'users_suspended' ...
Total time: 00:00:00.437
```

```
====================================================
Getting 'useruploads' table structure ...
Creating table 'useruploads' ...
Total records: 36
Inserting records to 'useruploads' ...
36 records inserted.
Creating index for 'useruploads' ...
Total time: 00:00:00.281
====================================================
Creating relations ...
Total time: 00:00:00.109
====================================================
Total Time: 00:01:00.000, migration completed!
```

List of Figures

List of Tables

Name: Hiba Bouzaida Matriculation number: 923781

Honesty disclaimer

I hereby affirm that I wrote this thesis independently and that I did not use any other sources or tools than the ones specified.

Ulm, ...

Hiba Bouzaida