Migration of a web service back-end from a relational to a document-oriented database

Sebastian Drenckberg[1], Marius Politze[2]

[1] IT Center RWTH Aachen University, Seffenter Weg 23, 52074 Aachen, drenckberg@itc.rwth-aachen.de
[2] IT Center RWTH Aachen University, Seffenter Weg 23, 52074 Aachen, politze@itc.rwth-aachen.de

Keywords
relational database, migration, document-oriented-database, MongoDB, LINQ, SQL

1. INTRODUCTION

IT infrastructure and applications are becoming more important to the universities' processes and employees and to students and their daily life. This leads to increased competition among the universities to present the best and most appealing services to their students. There are many examples of small-scale developments that support individual learning use cases.

Often these services are instantiated by a student or research project but have to be maintained by universities technical personnel on the long term. Without considering enhancements, maintenance costs alone can be high to keep the services operational and migrate them towards new technologies.

One of these services of Aachen University is an Audience Response System (ARS) used to support large-scale lectures with more than one thousand participants. The ARS is currently supporting more than 40 lectures and other events. As part of the application lifecycle, the technological basis needs to be migrated so the service can be continuously operated.

2. PROBLEM STATEMENT

In the current infrastructure, the database server poses a single point of failure. An off site backup protects from data loss but allows neither automatic fail-over nor scaling. Like many modern real time web applications, this scenario requires scalable application and database software architectures. Our goal therefore is to migrate the application to a more scalable topology that uses replication in order to distribute data store and access to all available nodes.

The database should thus be migrated from *Microsoft SQL Server* to *MongoDB*. Changing the database back-end, however, also affects parts of the application logic. Based on the example of the ARS, a general approach shows standard cases when migrating from a relational to a document-oriented database model. As there are many services using the infrastructure, the goal is to generalize these cases to develop guidelines for migration of these other services.

As a relational database, the current *SQL Server* supports the use of constraints. This means that there can be references between datasets in the database. Using an *ORM* further allows accessing the referenced datasets directly via the object model.

The *ORM LINQ to SQL* offers a programming interface that integrates seamlessly into the *LINQ* language extensions offered by the C# programming language and allows addressing relations and attributes directly from code.

3. DATABASE MIGRATION

Obviously, migration of the database engine also requires migration of the object mapper and database connection classes. It is however required that the general functionality of the migrated class structure retains the described properties like accessing with *LINQ* and compile time syntax checking. The database migration is therefore performed in three steps:

1. Analysis and simplification of the current database model:

Reducing the complexity of the database model and existing relations allows effectively utilizing the trades of the new database system.

2. Conversion of database connection and object models:
   Actually migrating existing data and also exsitng source codes to the new database models and drivers and trying to preserve most of application logic.
3. Validation and migration of existing data:
   Using automated unit and integration tests to compare the application behaviour before and after the migration.

4. GENERALIZATION

Based on the possible archetypes of relations (1:1, 1:n and n:m) general guidelines were formulated: these identified cases serve as a reference when other applications need to be converted. Document-oriented features such as embedding, referencing or using multiplicity of lists allow more efficient data structures than often modelled using relational databases.

For example, typical intermediate relations can be removed as it is possible that multiple references can be directly stored in documents. Furthermore, it is also possible to combine or remove tables by hierarchically integrating information from multiple relations into a single document.

Migration to a document-oriented database system additionally allows an optimization of the applications, since each document can be individually modified or extended in its structure. This allows applications to change the stored data more evolutionary and thus increases the overall maintainability of the software.

5. CONCLUSION

The migration of a web service back-end to a different database system should be well considered. Without a basic concept, it is not possible to successfully perform such a migration without service interruptions. Already existing structure of the database relations must be analyzed and adapted to the new system. Subsequently, optimization points of this structure can be characterized.

Looking at this first use case, the migration was very successful. Compared to the old technology stack, it did not only increase long-term maintainability of the software but also reduced overall resource consumption. The set of generalized guidelines further ease future migrations planned in the near future.

6. AUTHORS' BIOGRAPHIES

Sebastian Drenckberg, B.Sc. is software developer at the IT Center of RWTH Aachen University since 2017. In 2017, he finished his B.Sc. studies in Scientific Programming at FH Aachen University of Applied Sciences and his apprenticeship as a mathematical-technical software developer.

Marius Politze, M.Sc. is research associate at the IT Center RWTH Aachen University since 2012. His research is focused on service-oriented architectures supporting university processes. He received his M.Sc. cum laude in Artificial Intelligence from Maastricht University in 2012. In 2011, he finished his B.Sc. studies in Scientific Programming at FH Aachen University of Applied Sciences. From 2008 until 2011, he worked at IT Center as a software developer and later as a teacher for scripting and programming languages.