Master of Science in Engineering: Game and Software Engineering
May 2019

Automated Culling of Data in a Relational Database for Archiving

Simon Nilsson

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Engineering: Game and Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Simon Nilsson
E-mail: sina14@student.bth.se

University advisor:
Professor Håkan Grahn
Department of Computer Science

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background. Archiving of legacy information systems is challenging. When no options exist for extracting the information in a structured way, the last resort is to save the database. Optimally only the information that is relevant should be saved and the rest of the information could be removed.
Objectives. The goal is to develop a method for assisting the archivist in the process of culling a database before archiving. The method should be described as rules defining how the tables can be identified.
Methods. To get an overview of how the process works today and what archivists think can be improved, a number of interviews with experts in database archiving is done. The results from the interviews are then analysed, together with test databases to define rules that can be used in a general case. The rules are then implemented in a prototype that is tested and evaluated to verify if the method works.
Results. The results point to the algorithm being both faster and able to exclude more irrelevant tables than a person could do with the manual method. An algorithm for finding candidate keys has also been improved to decrease the number of tests and execution time in the worst case.
Conclusions. The evaluation shows results that point to the method working as intended while resulting in less work for the archivist. More work should be done on this method to improve it further.

Keywords: digital preservation, Information Systems, legacy systems, database reverse engineering

Sammanfattning

Bakgrund. Arkivering av gamla informationssystem is utmanande. När det inte finns några alternativ för att extrahera information på ett strukturerat sätt är siste möjligheten att arkivera databasen i sig. Det är då av intresse att endast den information som är relevant sparas. Resten av informationen bör tas bort.
Syfte. Målet är att ta fram en metod för att assistera IT-arkivarien i processen av att gallra en databas före arkivering. Metoden ska beskrivas som en samling av regler som definierar hur oviktiga tabeller kan identifieras. Förhoppningsvis kan resultatet fungera som en grund för framtida arbeten.
Metod. För att få en bättre förståelse av hur processen för att arkivera en databas funkar idag samt vad som kan förbättras genomförs intervjuer med experter inom databasarkivering. Resultaten från intervjuerna analyseras sedan, tillsammans med test databaser för att definiera regler för det generella fallet. Reglerna implementeras sedan i en prototyp där de kan testas och jämföras med den manuella metoden.
Resultat. Resultaten pekar på att algoritmen både är snabbare samt kan exkludera en större del av den oviktiga informationen än vad den manuella metoden lyckades med. En algoritm som användes för att identifiera kandidatnycklar i databasen kunde även förbättras. Förbättringen gav ett signifikant bättre resultat i det värsta scenariot.
Slutsatser. Resultaten från testerna visar på att metoden fungerar som förväntat, samtidigt som en sparar tid för arkivarien. Framtida arbeten bör göras för att vidareutveckla metoden.

Nyckelord: digital arkivering, informationssystem, legacy system, databas rekonstruktion

Acknowledgments

I would like to thank my colleagues at Sydarkivera, Mårten Lindstrand and Elin Jonsson, for their assistance with test databases and the tools I needed. I would also like to thank my supervisor Prof. Håkan Grahn for the helpful advice, support and encouragement through the thesis. Finally, I would like to thank the experts that took the time to let me interview them.

Contents

Chapter 1
Introduction

Archival information in the private and public sectors are facing big challenges in the near future to adapt to our digital world. In most cases the digital information is archived on paper because export functionality are missing in the systems [11]. At the same time the information generated is growing and therefore also the digital data that has to be archived. To make this more sustainable the data should and are therefore stored and preserved digitally. As an effect of the missing export functionality the systems are in many cases left online as inactive systems. This is very costly since both licensing and running costs have to be paid continuously.

1.1 Background

Many of the systems build upon some kind of database layer. This layer is in most cases based on SQL [11]. The information in the database consists of all the user data as well as logs and program data that might not be relevant for archiving. From an archival standpoint much of the data can be ignored as irrelevant and should optimally be removed before archiving. Removing the irrelevant data will both save space and make the relevant information more accessible. The task of removing the unimportant data is done manually which is very time consuming since the databases generally contain hundreds of tables [11]. The size of the databases will of course vary but an estimate made by Sydarkivera [11] suggests that the median size is around 300 tables per database. The problem is thus that we have access to a lot of data of varying relevance that needs to be sorted through and culled.

Culling is defined in this thesis as removal of information and structure from the database that is not needed in the final archive. The concept of what information is relevant and what information can be removed is of course difficult and hard to define. For this project we use the laws specified for each type of information. This means that for example tables describing a case in a system with names and notes from the case are important to save. A table that is empty is considered unimportant since it only provide information of the systems structure, which is not relevant. A table with sign in dates for the users of the system are often unimportant since it does not provide any information about the cases themselves. There are however exceptions that require interactions with the archivist. If the systems saves medical records, then the log tables must be saved a minimum of 10 years.

The national archives in Sweden is currently working on establishing specifications for data transfer called "FörvaltningsGemensamma Specifikationer" or FGS[21]. These specification specify how information should be saved for long-term storage.

There are currently an ongoing project that is defining how relational databases should be archived. The idea is that if the system can be archived by some export function in the system, then that is the preferred method. In the case that there are no export functionality in the system, the whole database should be extracted for archiving. Before final archiving, the database is culled from unimportant information and documented. This process will inevitably make the information harder to comprehend than if it were still present in the system, but that is a loss that is considered unavoidable since the systems cost energy and money to keep online. The resulting format will consist of a XML structure of one or multiple files defined by the ADDML [20, 13] or SIARD [24] standard. Sydarkivera are actively working on improving the readability of the information by documenting how the information was used in the system.

The current research have investigated how digital information can be preserved long-term. There are multiple papers describing different methods and architectures for digital preservation [1, 15, 3]. In the other end there are papers and tools where the relational database can be extracted and converted into an XML format [8, 7]. There are also much research done on reverse engineering the structure of a relational database when the relations is missing [2, 14, 25, 12]. What is lacking is a way to simplify the process that the archivists go through when archiving a relational database. By utilizing the tools of reverse engineering a database it is likely possible to simplify the manual work that is being done to cull the relational database before archiving it.

The work will focus on software engineering to improve the process of archiving a relational database. By developing a prototype based on interviews and an analysis of the needs the aim is to assist in a specific task, which is central in the field of software engineering. The prototype was also presented to users for feedback that did further improve the software during the process.

1.2 Aim, Objectives and Thesis Questions

The aim of this project is to optimize the process of archiving the information stored in a relational database. This project will focus on the culling step where the database is reduced, leaving only the important information. The culling process is cumbersome and takes a lot of manual work which makes it an ideal area for automated improvements.

RQ1 How are the information in relational databases prepared from running in the information system to a format ready for archiving today?

To be able to improve the way relational databases are archived it is important to first understand how the process works today. This can be done by questioning multiple different organisations with experience in the area. Their processes can then be collected and summarized which will give a better understanding of the problems and difficulties going into archiving and culling relational databases. By answering the question a good and efficient strategy can be decided for how the process can be improved. The objective is to get a good understanding of what can be improved and how to improve it.

RQ2 What general rules can be defined for deciding the archival value of a table in a relational database?

To be able to simplify and partially automate the process of culling a relational database it is important to define rules of what information have to be saved. The rules will be defined in such a way that they can then be used in an algorithm to remove the data with no archival value. The aim is to find structured rules defining all the types of tables and whether they should be saved or not. There are of course a near infinite number of tables that can be created so the rules have to be created to describe general criteria. For the cases where it is not always certain that a table can be archived or culled rules will be defined with the requirement of user interaction for verification. These rules can't contribute to a fully automated system but they will be valuable for an interactive system.

RQ3 How much will a system implementing rules for information culling speed up the archival process?

Since the overall objective is to improve the process of culling the relational databases it is interesting to see if it is successful. A tool will be created implementing the previously defined rules. The tool will then be used in parallel to the archival process of databases to evaluate its speed. The evaluation metric will be the time the prototype saves compared to the current process of manual culling.

1.3 Deliminations

The rules defined in this report will only consider what tables can be culled and not if specific columns can be culled. It is clear from the database analysis that some columns could easily be removed to further reduce the size of the result. This delimination is done to be able to complete the project in the limited time.

The rules will not be based on the individual information types, but rather defined as general rules that can be applied on all of the information types. It is likely more efficient for a specific information type if rules are specifically specified for that type, but it requires much more time and work to specify unique rules for each type. By specifying some general rules that can be applied to all types a limited but still helpful method can be developed for all types.

1.4 Contribution

The report presents a proposed method for reducing the time it takes for an archivist to cull a database for final archiving. The method is also reducing the size significantly of the archived database. The result should be viewed as a base for future research in the area.

An improved algorithm for finding candidate keys in an arbitrary database is also presented, based on an algorithm by [2]. The new algorithm reduces the worst case significantly which in turn makes the analysis run in reasonable time on real world databases.

Chapter 2

Related Work

To be able to cull an existing database, the first step is to gather information about the structure and content of the database. There is multiple research papers on the subject of reverse engineering a database. The terminology varies depending on source. This report will use the definitions created in [4].

One important step in understanding the structure of the database is to first extract a model that is easier to process and understand. This is discussed and described in multiple methods [23, 9, 29]. There are also literature describing ways of breaking down the generation of the meta-model into smaller clusters which can be analysed and processed one by one [26, 27]. These methods do however all require the database schema to be present with all the structural information of the database. Most of these methods do also require human assistance to extract the information and can therefore not be automated.

Another useful method for reverse engineering a database is to use the database schema to extract inheritance from the database [10]. Inheritance is a concept that is not present in the relational database schema but it is commonly used on the software side. By extracting the potential inheritance from the database more information of how the data is used can be reverse engineered. The algorithm requires the database schema and the data to analyse it using a combination of algorithms.

There are however other alternatives that can be used when the database schema is incomplete or lost. By analysing the data in the database it is possible to extract the structure of the database which can then be used in further analysis [17]. The approach can be described in short by first finding the candidate keys in the database based on the content of the tables. The candidate keys are then compared and tested against all the other columns to find the relations. An improved and clarified version exists where the extraction is described as concrete steps [2]. These steps can be used to extract the candidate keys, foreign keys as well as the cardinality of the relations in the database with no human intervention. The algorithm works by systematically analysing the information in the database to reverse engineer the schema based on what would make it valid as an SQL database. The algorithm makes no assumptions about the database and only uses the information available in the database. Other methods use the code or the application views to extract the database schema, which could yield better results while requiring a person to perform the reverse engineering process [14, 25, 12].

Another technique that might be useful in making an archivist understand the content of a database could be to visualize it. One method that is created to make the database model understandable to the end user is to generate a form based UI application from the database [19]. The technique generates an application where the user can see an approximation of the final layout directly from the database. Forms are considered easier to understand than a entity-relation model in their tests.

There are little research done in the preservation of databases that could be used to assist in this problem. One article describes how archiving and digital curation is done today, while also describing the areas that need further work [16]. The information in the article is focused on the more archival aspects and very little on the technical details. The conclusion is that we are only in the early stages of automatic digital curation and archiving.

Using data analysis for archiving and curation have done before, but with a different purpose. One algorithm describes a way to data-mine the articles and assist the user in finding the important metadata [22]. This is supposed to speed up the process of curating the articles before adding them. A similar method could be used when assisting the archivist in the archiving process of a legacy database.

To preserve the databases long-term other data formats are used such as XML [18] where the database can be converted into a format that is not dependent on any specific system or program. Archiving digital information is complex since digital systems are far from stable. The paper suggests that many types of digital object rarely survives past 5 years until it is replaced by newer systems. By defining an XML structure called DataBase Markup Language (DBML) the content, structure and attributes can be preserved past the lifetime of a RDMS.

The relevance of solving the culling problem is largely based in the archival domain. It is however a problem that most of Sweden's municipals and public authorities have or are going to encounter in the next few years [21]. Similar problems will and have also arisen in many other parts of the world. There are many ongoing EU-projects that are investigating improved methods for archiving data, E-ARK4all [6] among others. There are also an organisation called DLM-forum [5] that collects tools and methods for digital archiving in which Sydarkivera is part of.

Chapter 3

Method

To be able to answer **RQ1** a series of *semi-structured interviews* is made. By performing *semi-structured interviews* with experts from multiple organisations on their methods for archiving databases the problems and limitations can be summarized. The candidates for the interviews are people who have worked with archiving databases at a technical level while still having knowledge and understanding of archiving. It is important the interviewees have experience in the area to make the results more relevant. The goal is to get a better understanding of how archiving of databases is done in the different regions and municipalities in Sweden. It is especially interesting to see if they have used any tools to simplify the process or if it is all done by hand. If they have any ideas of what could be automated, it will be used for building the prototype. An interview is chosen over forms since it is expected to yield better results and the time will be Negligible since the number of people doing this kind of work is limited. To get the best possible results from a limited number of interviews it is important to have a good spread of organisations in the interviews. In reality the spread was much higher since all the persons interviewed had worked in multiple organisations. The interviews will follow the questionnaire in appendix A.

The results from **RQ1** was used and the suggestions made was verified in a *comparative evaluation* to find the answer to **RQ2**. By comparing the manually culled version and the original version of multiple databases the patterns from the interviews can be verified and new patterns can be found. The patterns should be defined as rules and criteria of what information can be saved and what information can be culled. Information that can't be placed in either absolute group will be defined with rules describing the level of validation needed from an archivist. The contents of the tables will be used to decide if the table is relevant, but the culling itself will always be on a per table basis.

Finally, to answer **RQ3** a prototype is built, based on the result from **RQ1** and **RQ2**, which is then tested in an *experiment*. The prototype is built implementing the rules and criteria defined in the previous two steps. The prototype is evaluated based on its ability to perform the same task as a person would do today. The tests will be done by letting a user, experienced with the manual archiving process, archive a set of test databases so the methods can be compared. The test person will alternate the order of the methods, starting with the prototype in half of the cases and the manual method in the other half. To avoid bias as much as possible, the two tests of the same database will be separated with other tests.

Three metrics will be considered in the tests as well as any comments the user

Table 3.1: Metrics that will be measured in the experiment

Metric	Motivation
Duration	The duration in time the archivist spend with the culling process. Measured in minutes.
Number of tables	The number of tables left in the database that should be saved. The goal is to remove all the unimportant tables so the fewer tables left the better, as long as no important tables are removed.
Important tables removed	It is crucial that no table that containing important information are removed since the goal is to preserve information.

have during the tests as described in table 3.1. The first metric is the time it takes for the archivist to perform the culling process. Only the actual work performed by a human will be measured, leaving out the time the prototype does its automatic analysis. This is because the cost in an organization is the work time spent on the extraction. Therefore, the time it takes for the application to complete its analysis can be excluded since the user can do something else during that time. The second metric is the amount of tables left after the extraction. The fewer tables that can be saved, while still keeping all the important information is better. It is expected that the manual method will contain only the necessary tables, so the closer to the manual result the prototype can reach the better. The result will then be compared to the information that is known from the delivery decision to be saved. If all the information in the delivery instruction can be found in the result of the process, then it is considered a success. If, however, there are tables missing with important information it would be considered a fail for the algorithm.

Chapter 4

Results of Research Question 1

This chapter presents the results for **RQ1**. The results will then be used in section 5.1 where the results for **RQ2** are presented. The results of both will then be used to create the prototype that is used in the experiment in chapter 6 which presents the results for **RQ3**.

The interviews represent a good spread in the public sector in Sweden as seen in Table 4.1. The five interviews gave us information of how archiving of databases is done in different organizations. The results were consistent between all the participants. Each participant who are present in multiple rows in the Table 4.1 has worked in different organizations previously with the current on top. Their experience can be seen in Table 4.3 where the average work experience is just over 9 years, all with multiple database extraction projects performed. Some participants did however not have exact numbers for how many databases they had archived and their numbers are therefore approximated by them. A larger summary of all the results of the interviews can be found in Table 4.4.

4.1 Archival Process

The process always starts with an investigation from the archivist of what information needs to be saved. This investigation is not based on what information that can be extracted but rather an analysis of what information was entered into the system that needs to be archived. The investigation is done by an archivist who know what the

Table 4.1: A list of the spread of the interviewed persons

Participant	Workplace
A	Sydarkivera
B	Sydarkivera Region Skåne
C	Region Skåne Trafikverket Region Östergötland Landskrona stad
D	ESSolutions Stadsarkivet Göteborg
E	Arkivnämnden Göteborg Region Västra götaland

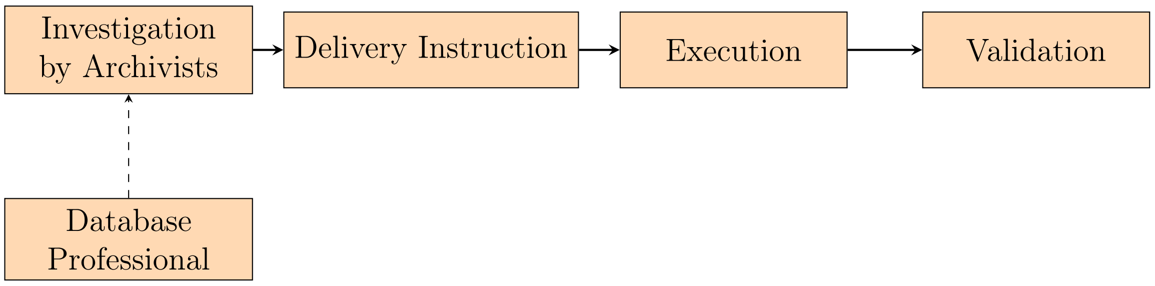


Figure 4.1: The workflow for archiving a database

laws and practices on the information-type the information system holds. Sometimes this investigation is assisted by database professionals to make an initial analysis of the actual database to accompany the archivists knowledge. The results of the investigation will then be used as a basis for the actual data extraction specifying what needs to be archived. When the initial investigation is complete the next step is to investigate how the information can be extracted. The participants all described that the choice of extraction method starts at the end, namely with how the information will be used in the future. If, for example, the information-type is patient journals then they would ideally be used as journal documents rather than XML or as an archived database. This would enable future request to be delivered just as it would be delivered when the system was still in use. It is however not certain that it is possible to extract the journals but it is preferred when possible. Then, the next best alternative is investigated and so on. In some cases there are already builtin functionality that extracts the information in an acceptable XML format. In other cases the information in the system is used for statistics where an extraction of the whole database can be the most convenient method. The final case is when there are not option to extract the information in an usable format so the database is extracted and documented.

When a method for extraction has been decided the results are compiled into a delivery instruction. The instruction contains what information should be saved and what format it should be saved in. According to the interviews the delivery instructions could contain as much as a list of the tables that should be extracted or as little as a list of the general information that should be extracted. When all the investigation is done the extraction can start. This process can either be done internally in the organization or by hired consultants depending on their expertise. When the extraction is done it is compared and verified against the delivery instructions. An overview of the process can be seen in Figure 4.1.

4.2 What can be Improved

The most time-consuming tasks in this process are the investigation of what information is required and, if no builtin option is available, the extraction of the information from the system. The first step in the process is case dependent and no-one in the interviews thought it could be automated. Most of the interviewed candidates were however convinced that some automated improvements could be done in the information extraction phase, especially when the whole database is extracted. One of the

Table 4.2: Rules suggested in the interviews

Rule	Should be removed	Need verification	Can be automated
Empty tables	Always	No	Yes
Empty columns	Always	No	Yes
Tables with few rows (less than 10)	Often	Yes	Yes
Tables containing system information	Always	Yes	Maybe
Tables containing system logs	Mostly	Yes	Maybe

interviewed candidates had even built a simple tool to assist the process by running a program that could summarise the content of a database.

The interviewees suggested that empty tables and empty columns could be removed in every case they had encountered. It was also possible to remove columns and tables that contained very little information. This was however not always the case and manual verification was required before it could be removed. Tables containing application information like value lists could be deleted when it was not related to any other table with relevant data. These values were likely used in the actual system but since they have no relation to the rest of the information it has no value for archiving. Tables containing system logs like the list of all user sign-ins, could be removed most of the time with some explicit exceptions where it is required that the logs are saved a specific number of years. The rules for if they should be saved or not are however known to the archivist and can therefore be used as a parameter. These suggestions are presented as rules in Table 4.2.

It was clear that there are a lot things to be improved in the whole process especially since none of the interviewees had used any specialised tool to assist them except the internal tool that was used in Gothenburg. Everyone had used SQL Server Management Studio from Microsoft, sometimes in combination with scripts in python or bash. The sizes of the databases varied depending on what type of information it contained. It was however possible to get an average of about 250 tables per database with an average storage size of 1 gigabyte. A summary can be seen in Table 4.3.

A tool that implements these rules could help the archivists understand the relevant contents of the database better. If the tool also visualizes the information with focus on the archival process this would likely further decrease the time it takes to archive it. The rules that are more ambiguous can be displayed so that the archivist can verify them while the ones which are not can be removed or included automatically. There are likely more rules that are not yet discovered that can be found if the databases are analysed.

Table 4.3: Experience of each person

Participant	Experience	Number of databases archived	Size of databases
A	5 years	10	300
B	10 years	6	250
C	6 years	20-60	300
D	9 years	10	200
E	16 years	30	200-300

Table 4.4: Summary of interviews

Participants	A	B	C	D	E
Where do you work?	Sydarkivera	Sydarkivera, Region Skåne	Region Skåne, Trafikverket, Region Östergötland, Landskrona stad	ESSolutions, Stadsarkivet Göteborg	Arkivnämnden Göteborg, Region Västra götaland
What is your field of work?	head of IT and digital archive	IT archivist	Information management	Preservation strategy	Digital deliveries
How long is your experience of archiving information?	5 years	10 year	6 years	9 year	16 years
How long is your experience in database?	1 year	1 year	6 years	9 years	16 years
How many databases have you archived	10	6	20-60	10	30
Haw large was the databases? (rows)	300	250	300	200	200-300
What DBMS was the databases based on?	Microsoft SQL server, Microsoft Access	Microsoft SQL server, Microsoft Access	Microsoft SQL server	Microsoft SQL server	Microsoft SQL server, Oracle SQL, Informix, DB2
What was the most time consuming aspect of the archiving process?	Investigation and execution	Investigation by archivists	Execution	Investigation	Investigation
What tools was used in the extraction process?	SQL Server Management Studio	DbVisualizer	SQL Server Management Studio	Custom built excel summary tool	SQL scripts and Microsoft Access
How do you think the process to cull a relational database for final archiving can be improved?	By sorting and presenting the database to make it easier to comprehend	-	-	Better visualisation	By planning how to archive the system before it is ever used
Have you found any patterns in the databases you have archived?	Transaction tables, index tables and empty tables can be removed. Small tables tend to be less relevant	Logfiles can be removed depending on the age of the system	-	Lists of values, empty columns and log files can be removed	Start from the main tables and add everything that is related and relevant

Chapter 5

Implementation

In this chapter the results from **RQ1** is used to define general rules for which tables can be culled, thereby presenting the results of **RQ2**. It will also describe the implementation of the rules into a prototype which is used to answer **RQ3**. Section 5.1 uses the results from **RQ1** to define a set of general rules that can be used to cull a relational database for archiving. The section will therefore present the results for **RQ2**. Section 5.2 describes how the prototype works and how it uses the rules. Section 5.3 describes the improvement that could be made to the algorithm used for finding candidate keys.

5.1 Database Analysis Based on Interviews

It was clear from the start of the thesis project that there existed many empty tables in the tested databases. There is no need to know anything about the database schema for the purpose of archiving. Therefore tables that do not contain any rows can be removed. The reason behind the number of empty tables in the test data can be explained by studying the usage of the system. The systems tested have all been large and contained multiple features that have not been in use by the organization using the system. This confirmed what the interviews revealed about empty tables. Although it is out of scope for this project it was clear that there was a number of empty columns as well that could be removed. Of all the relational databases that were analysed 53.0% of the tables were empty.

The interviews revealed that tables containing system logs and system information could be removed in most cases. These are however very difficult to identify by a generic algorithm. No method for confidently finding the system information and logs could be found and the rule is therefore not considered for the implementation of the prototype. One subset of the system information tables could however be identified.

Tables that only contain one column in the databases tend to contain information that existed in a menu in the system. Since the table does not contain any link to the user entered data the information it is not relevant. Other tables that contains two or more columns usually link a foreign key to a human readable value are important to save, but when there only exists one column the table don't contribute any information. An example of such a table can be seen in Table 5.1. These tables can almost always be removed automatically, but since it is not certain the user needs to verify it.

A variation of the tables with one column are the ones with two columns linking

Table 5.1: Example of table with one column

Funktion
FÖRSLAG
PROTOKOLL
SIGNATUR
SÄNDLISTA
UNDERLAG

Table 5.2: Example of table describing a foreign key

atgard_riktning	atgard_rikt_text	aktiv	mod_dat
I	Inkommande	J	2005-04-27-11.24.34.840000
U	Utgående	J	2005-04-27-11.24.34.840000

a foreign key with a value. They can be identified by a candidate key that links it to other tables while only containing a maximum of about 20 rows and less than 5 columns. It is common that a table in the database is used to describe a simple value in more detail, like Table 5.2. In this case the letter 'I' and 'U' are used for 'Inkommande' and 'Utgående'. This would be valuable to save if the value 'I' and 'U' are used anywhere else in the database. If however the value is not used anywhere, or if the table that uses it can be removed this table can also be removed since it does not contribute any information by itself. However this rule can't be used in every case. To make sure the quality of the result is good enough these tables should preferably be verified by the archivist. The easiest way to identify these tables is to automatically search the database for tables with content but with no relations to other tables.

By analyzing the content and structure of the databases it is clear that most of the data that is relevant is connected by foreign keys in the database. By analysing the relations a selection of disjoint sets of tables can be identified. The largest set is likely the set that should be archived which in turn can be verified by the archivist. The smaller sets are likely not relevant for archiving, which once again have to be verified by the archivist. Identifying all these sets and letting an archivist verify if they can be removed will decrease the time the archivist have to spend on the extraction process. An example of such sets can be seen in Figure 5.1. The tables *User*, *Case* and *Environment* are all related to each other and are therefore likely to be important. The tables *Sysinfo* and *Unit* are also related, but in a smaller set only containing 2 tables. Since *Sysinfo* and *Unit* is the smaller set the tables should be marked as unimportant and verified by an archivist. In the test cases the largest set contains an average of 73% of the tables with the reset of the tables split into sets with just a few tables each. The tables that are in the smaller sets are presented to the user for validation before they can be removed.

Table 5.3 lists the rules that are concluded to be possible to implement in an automated system while not compromising the integrity of the information in the database. Some of the rules from Table 4.2 have been removed since they are too ambiguous while others have been added based on the structure of the databases. These rules are the results of **RQ2**.

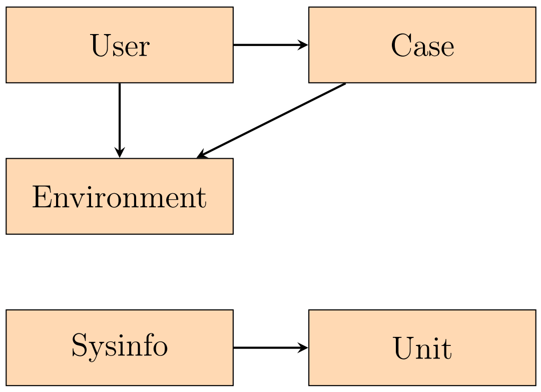


Figure 5.1: The relations of tables showing different sets of tables that are linking

Table 5.3: Rules suggested extracted from analysis

Num.	Rule	Should be removed	Need verification	Can be automated
1	Empty tables	Always	No	Yes
2	Empty columns	Always	No	Yes
3	Tables with few rows (less than 10)	Often	Yes	Yes
4	Table with only one column	Often	Yes	Yes
5	Tables with no relations	Often	Yes	Yes
6	Tables in the smaller sets	Often	Yes	Yes

5.2 Prototype

To test and verify that the archiving process can be improved with a tool that automates the initial analysis, a prototype is developed. The prototype is built as an application that can be run on any computer. It is connected to the database remotely or locally and allows for easy browsing of the database. The application is built as an Electron application in Javascript using the framework React for UI rendering. A Nodejs library called MSSQL is used to connect to the server and all the schema information loaded from the database is stored locally with MobX.

When it is connected the archivist can start an automatic analysis which takes between 10 and 60 minutes to complete, depending on the size of the database. This analysis performs multiple different steps analysing the database and extracts a list of all the tables that can be removed according to the previously defined rules. Structure information that is missing from the database is recreated as far as it is possible to allow for a more accurate analysis. The whole analysis is described in Figure 5.2. In short, the algorithm fetches the database schema from then DBMS and then calculates the candidate and foreign keys if they are missing. Then the rules are applied to find tables matching them. For rules where it is not certain that the table can be excluded it is instead saved into a list along with why it was identified. This list should later be used as a starting point for the archivist to immediately exclude a part of the database.

Once a database is selected an overview of the contents of the database is presented as seen in Figure 5.3. Information of the database composition such as the

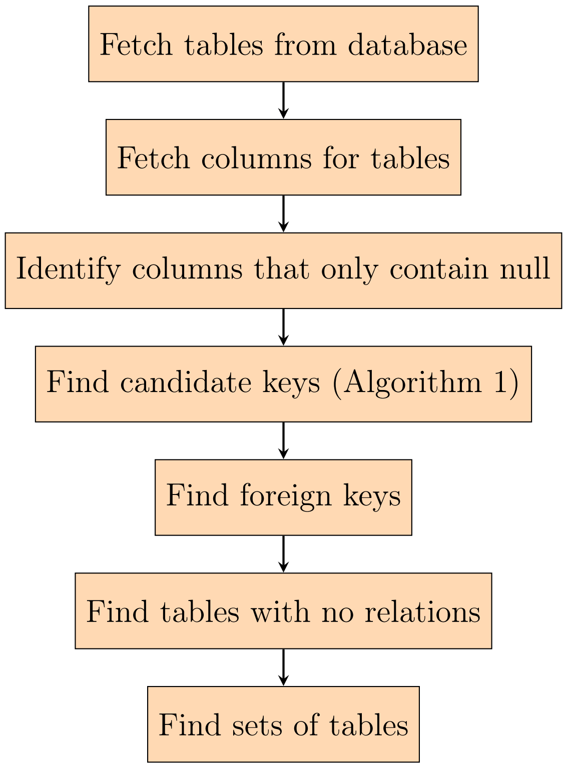
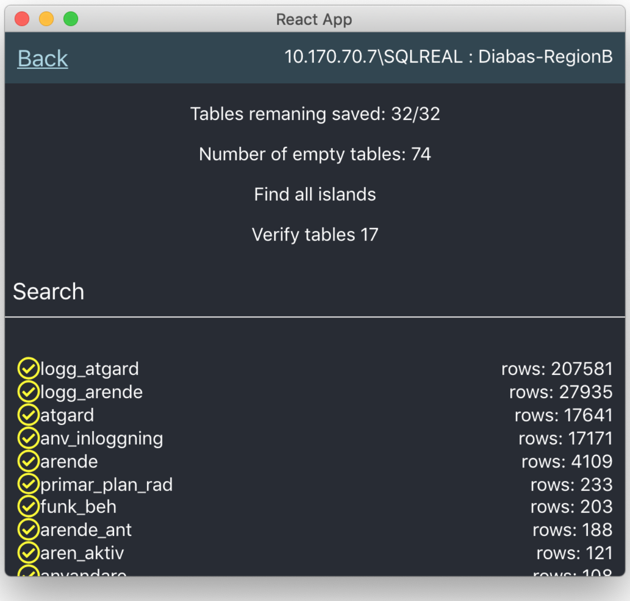

Fetch tables from database
Fetch columns for tables
Identify columns that only contain null
Find candidate keys (Algorithm 1)
Find foreign keys
Find tables with no relations
Find sets of tables

Figure 5.2: Application analysis

number of empty tables as well as the number of tables that are still left to bee saved is displayed. When the initial analysis is completed the button "Verify tables" appears, where all tables the algorithm have found to be unimportant are listed. The tables are grouped based on why they was selected. Every time a tables is listed it is possible to click the table name to navigate to that table and get more information about it. This view is shown in figure 5.4 where a table with only one column is presented. It is possible to navigate among the rows of the application as well as seeing the tables that are related to this. The yellow checkbox is used to select if the table should be included or removed. Yellow means that it is unknown, which will automatically mean it is saved. If it is pressed it will turn to green (save) and then red (remove).

5.3 Foreign Key Analysis

The algorithm used for finding foreign keys when they was not present in the database schema is a modified version of the algorithm Alhajj [2] presented in his paper. The algorithm consists of two parts. First all possible candidate keys have to be found, then they will be used to test all possible combinations of columns which could be pointing to that candidate key. The first part of Alhajjs algorithm is based around the definition of candidate keys. A candidate key is defined as *the minimum set of attributes that uniquely identifies each tuple in a given relation*. The algorithm therefore tests all possible combinations of the columns to see if it is a possible candidate key. This is done by selecting all distinct values in the selected columns


React App

Back 10.170.70.7\SQLREAL : Diabas-RegionB

Tables remaning saved: 32/32

Number of empty tables: 74

Find all islands

Verify tables 17

Search

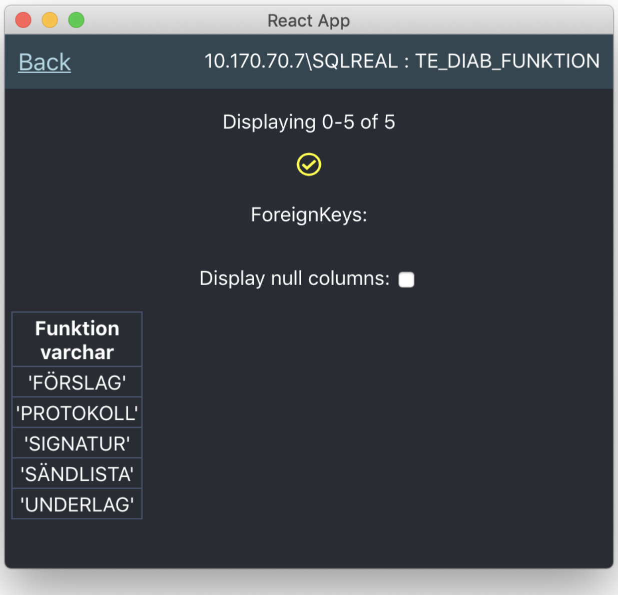
✓ logg_atgard	rows: 207581
✓ logg_arende	rows: 27935
✓ atgard	rows: 17641
✓ anv_inloggning	rows: 17171
✓ arende	rows: 4109
✓ primar_plan_rad	rows: 233
✓ funk_beh	rows: 203
✓ arende_ant	rows: 188
✓ aren_aktiv	rows: 121

Figure 5.3: The overview page of a database

and checking if they are equal in amount to total number of rows in the table. If the value is equal a candidate key is found since every row in the table have a unique value in the selected columns. This requires a lot of tests to be made, especially in the worst case scenario where all the columns together is a single candidate key. The worst case scenario is however not common in reality, but during the initial usage of the algorithm in the prototype multiple tables had more than 50 columns. Testing all the possible candidate keys in those tables turned out to be time consuming so an optimization had to be made.

The definition for candidate keys specify that a candidate key is the minimum set of columns that uniquely identifies each row. Selecting all distinct rows in the table must equal the number of rows in the table if there are any candidate key. This simple test can eliminate a lot of testing with the naive method proposed by Alhajj. It is however possible to improve the algorithm even further by using the definition again. A candidate key is the *minimum* set of columns, therefore all the columns together is not a candidate key if any of the columns could be excluded while still resulting in the same amount of distinct rows. This test can be repeated, eliminating column after column until only the minimum number of columns remain. The remaining columns should all be tested again to fin potential multiple candidate keys in the table. The algorithm is presented in Algorithm 1, which reduces the duration of the step.

There was not much that could be done to improve the algorithm for finding the foreign keys so it is left as Alhajj describes it. Finding the foreign key is therefore the most time consuming step in the whole analysis. One unfortunate result from this automatic analysis is that some potential relations that are not actual relations are discovered. To avoid this the option to only include foreign keys where the column



10.170.70.7\SQLREAL : TE_DIAB_FUNKTION

Displaying 0-5 of 5

ForeignKeys:

Display null columns:

Funktion varchar
'FÖRSLAG'
'PROTOKOLL'
'SIGNATUR'
'SÄNDLISTA'
'UNDERLAG'

Figure 5.4: Interface displaying a table

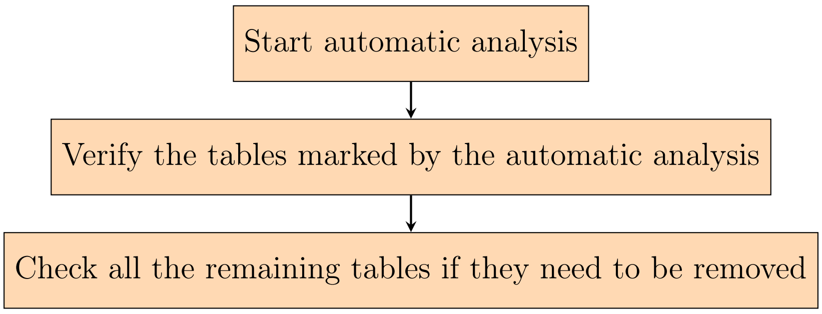

Start automatic analysis
Verify the tables marked by the automatic analysis
Check all the remaining tables if they need to be removed

Figure 5.5: User workflow

names match at least partially is added. In all the test databases this turned out to be an accurate way to only include the relevant relations. Some databases might not have as similar column names which would make this less efficient. The comparison should not only test if the column names is the same but rather similar. To solve this in a simple jet efficient way the jaro-wrinkler distance is used [28]. A distance threshold of 0.8 resulted in most of the false positives removed while saving all the actual relations.

```
Input  : A table $R$ and $P(R)$, the list of all columns in $R$.
Output: All possible candidate keys of $R$ are added to the CandidateKeys
        table.
let A = $P(R)$;
while size(Select distinct A from R) == size^1(R) do
    let $B = A$;
    for every column in $B$ do
        let $C = B$;
        C.remove(column);
        if size(Select distinct C from R) == size^1(R) then
            $B = C$;
        end
    end
    Add $B$ to CandidateKeys;
    $A = A - B$;
end
```

Algorithm 1: Improved algorithm for finding candidate keys

Chapter 6

Results and Analysis

This chapter presents the results from **RQ3**. The prototype implemented and tested in the experiment described below are based on the results from **RQ1** in chapter 4 and **RQ2** in chapter 5. The experiment was done with 4 databases, both using the prototype and using the manual method, in total, 8 test cases. It will give an indication of whether the method is feasible and therefore motivate someone else to investigate it further.

6.1 Prototype

On average 29% of the tables saved by the manual method could be identified as unimportant with the prototype. The prototype did exclude all the tables that the manual method excluded. This means that the result from the prototype is a subset of the result from the manual method. All the tables that was excluded while using the prototype was manually checked afterwards to find out if the prototype did fail or if the tables really could be excluded.

On average the prototype reduced the time the extraction takes by 35% saving the archivists time. All the test resulted in an equal or faster time for the prototype as seen in Figure 6.1. The smallest databases barely differed in time with one database having the exact same time of 11 minutes in both cases. However, the largest database showed a difference 82 minutes which means the duration was reduced by 75% compared to the manual method.

The results are positive. Every test that was performed showed that working with the prototype resulted in fewer tables saved than it did in the manual cases. This was a surprising results since it was expected that the manual method would result in the smallest possible set of tables that needs to be saved. It turned out the tables that were excluded all contained some data, and were excluded by the algorithm. In all cases the table that differed in the result was excluded since it had no relations to other tables, or because it only related to a small set of tables, following rules 5 and 6 in Table 5.3. The check could conclude that the extra tables that was excluded with the prototype was not necessary. The reason behind the manual method included them was simply because the archivist would rather save too many tables than too few. If the archivist can not confidently know if a table is needed or not, it is better to save it. The prototype could however provided more information to the archivist which helped in making a better decision.

There was only a small difference in the time it takes for the archivist to complete the process manually and with the prototype. One possible reason for this is that

Table 6.1: Results from analysis test

Database	Initial number of tables	Number of empty tables	Duration (minutes)		Number of tables remaining	
			manual	prototype	manual	prototype
A	106	74	22	17	25	14
B	490	356	110	28	125	93
C	43	19	11	11	22	19
D	71	18	28	16	50	34

the amount of empty tables slows down the archivists since the manual method does not sort those out. In the case of database B there were 356 empty tables that the archivist had to go through to see that they are empty. With the prototype on the other hand the empty tables are already eliminated.

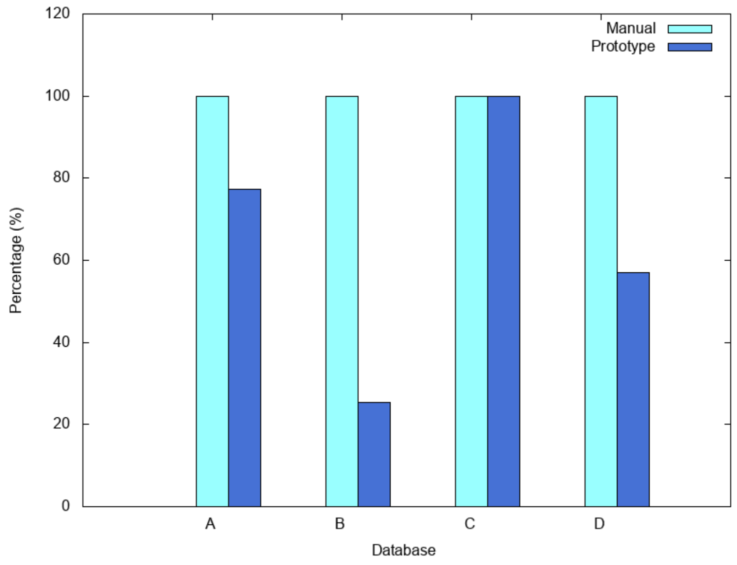


Figure 6.1: Time comparison of manual and prototype methods in percentage

During the tests a few questions and recommendations for improvements was discovered. When previewing a table in the prototype the top 30 rows are displayed with the possibility to view more if needed. There are however no option to view the last n tables which appeared to be useful to the archivist. This simple visualisation should probably be added to help the archivist making a better decision. There were also a request for a function where the archivist can search the whole database for a specific term. This could be used to easier find the most important tables by searching for the text that is entered into the system or text that is known and should be exported.

Table 6.2: Minimum and maximum number of projections of the suggested algorithm compared to the old algorithm

Number of columns	Earlier minimum	Proposed minimum	Earlier maximum	Proposed maximum
5	5	5	32	15
10	10	10	1024	55
15	15	15	32768	120
50	50	50	$1.126 * 10^{15}$	1275

6.2 Find Candidate Keys

The algorithm proposed by Alhajj[2] is useful in extracting the full database structure with all relations given no human intervention. It is however not efficient when handling tables with many columns since it is almost random in its testing order. This is largely the reason why a more structured method of testing can give such a large improvement in the amount of projections made. The minimum and maximum number of projections performed by Ahajjs algorithm are $\sum_{i=1}^{n} c_i$ and $\sum_{i=1}^{n} 2^{c_i}$, respectively, where n is the number of candidate keys and c_i if the size of each candidate key. The suggested Algorithm 1 have the same minimum while having $\sum_{i=1}^{n} i$ as the maximum. The maximum number of projections is reached when every column is by itself a candidate key. This differs from Alhajjs algorithm where the best case scenario is the same as the suggested algorithms worst case scenario. The minimum number of projections in the suggested algorithm is reached when there are only one candidate key. The result is an algorithm that is equal in the number of projections in some easy tables while resulting in an exponentially faster algorithm for the larger and more complex tables. An example of the number of projections for different tables can be seen in Table 6.2. The resulting candidate keys found by both algorithms are however always the same.

Chapter 7

Discussion

The results are interesting, especially since it was previously considered that the data extracted by the manual method was all the information that was required. It seems that the previous assumption was based on the lack of knowledge in the structure of the databases. It was clear after the tests that the information that could be removed with the prototype was indeed unimportant. It was however not clear until the database scheme was reverse engineered so that the archivists could get a better understanding of the structure. This by itself can be considered a good result since it suggests that there exists a method to remove even more unnecessary information.

The goal is however not only to reduce the number of tables that is saved but also decrease the time it takes for the archivist to complete the extraction. This was likely done by using the proposed method although not in every case. The time difference was more clear in the databases where the user had to go through multiple empty tables, which in itself are time consuming. It could have been even more interesting to know how long time the database had taken to extract if all the empty tables was already removed. That test would however require more tests to give any good indication if it does matter. Given the difference in time for the database B it is still possible that the prototype with the rules implemented does help in decreasing the time it takes to some degree.

The improvement on the algorithm for finding candidate keys was unexpected but necessary to be able to perform the analysis on the test databases. The algorithm presented by Alhajj is likely good enough in most cases, but it does not work in all of the databases that was tested. The most projections that it got stuck on in the initial tests was around 10000 which is not feasible, especially since every projection means on SQL request to the server. It is however suggested in his article that it can be improved by doing some limitations on the test set, but not to this extent.

There are multiple steps taken to improve the internal validity of the experiment. The method that was first tested on each database was switched so that half of the databases were first extracted with the manual method and half with the prototype. To reduce bias the same database was not extracted with the different methods directly after each other. Instead the order was specified to allow for multiple other databases to be extracted between the different methods. The knowledge of the initial system was specified beforehand so that the participant had the same amount of information in the beginning of each test. There were factors that ideally would have been improved but external factors limited the possibility. For example, the internal validity would have been higher if the experiment was performed with more participants.

The external validity was also considered when choosing the databases for the test. To get the best result, all the databases that could be gathered was tested. To improve the external validity databases of different information types were specifically collected for the tests to make the results as generally applicable as possible. Ideally more databases from even more different information types should be tested, but it was not possible in the given time frame.

Chapter 8
Conclusions

In this thesis, the process of archiving relational databases in the public sector of Sweden have been studied. The resulting information could then be used in the creation of a prototype with the purpose of simplifying the work the archivist have to do to archive the database.

The interviews provided valuable information that answers **RQ1**. The process of archiving a database from an information system can be split into four parts. Every step in the process consists of manual work with little tools that can assist. The typical workflow starts with an investigation where the archivists document what type of information should be saved. This documentation is then used to create a delivery instruction which specifies more specifically which information that is required. This is then used to perform the actual extraction. The last step is a validation that all the required information actually exists. Since there are very little tools to assist in the process it often takes much manual time for the archivists.

Based on the interviews a set of rules could be established answering **RQ2**. The rules specify that small tables and tables with no relations rarely contain valuable information and can therefore be removed. Another rule specify that tables with only one column does in most cases not contain any valuable information either. The concept of defining general rules for what patterns of tables that can be excluded in an export can definitely assist the archivist in the export process. It is, however, not feasible that this kind of general rules can be used alone. The rules suggested in this thesis could ideally accompany the export process along with other tools for better visualization of the relevant information and the opportunity to search the database with full-text.

The rules from **RQ2** could all be implemented in a prototype which could then be tested in an experiment to answer **RQ3**. The prototype did result in a smaller and more focused set of information being saved while still including all the important information. It does also save the archivists time by highlighting the important information so that the archivist can make better decisions. In the tests an average of 35% of the time the archivists spent on extraction could be saved and an additional 29% of the tables could be removed. There were not enough test-cases performed to answer the question conclusively and more testing have to be done to verify the findings. The feedback from the organisations that have used the prototype have been positive and they are continuing to use and develop the prototype.

A more efficient algorithm for finding candidate keys in a table was also presented which improved the worst case while not affecting the best case performance.

Chapter 9

Future Work

It would be interesting to see if the proposed method shows the same results in bigger datasets. As future work it would be interesting to see a larger set of databases that the tool can be tested on to give better prove the value of this method. It would also be interesting to compare the results between different data types to see if it works better in one or another. The idea was to make it general enough to work with any database but it would most likely not be the case for every database.

Another improvement that would be interesting to see is if the proposed method could be improved by combining it with other methods. The interviews and testing for example did suggest that a way to search the whole database for a text string could be useful to easier locate the most important tables. If the archivist easily could identify the main information an algorithm could start from that point and suggest other important tables based on it. The result would be a method which would work the other way around by including tables rather than excluding them.

It could also be interesting to see how much of the tables that are saved is really needed. During the investigation it was clear that many tables contained columns with very little important information. These columns could possibly be removed and further shrink the resulting information that have to be saved.

Finally a combination of both the general algorithm used here and more content specific solutions would be interesting since it would likely be able to be the best of both options. Is is likely that some information types can be culled much more efficiently when there are specific rules for that type.

References

[1] Kofi Koranteng Adu, Luyande Dube, and Emmanuel Adjei. Digital preservation: the conduit through which open data, electronic government and the right to information are implemented. *Library Hi Tech*, 34(4):733–747, 2016.

[2] Reda Alhajj. Extracting the extended entity-relationship model from a legacy relational database. *Information Systems*, 28(6):597–618, 2003.

[3] Christoph Becker, Hannes Kulovits, Mark Guttenbrunner, Stephan Strodl, Andreas Rauber, and Hans Hofman. Systematic planning for digital preservation: evaluating potential strategies and building preservation plans. *International journal on digital libraries*, 10(4):133–157, 2009.

[4] Elliot J. Chikofsky and James H Cross. Reverse engineering and design recovery: A taxonomy. *IEEE software*, 7(1):13–17, 1990.

[5] Dlm forum. `http://www.dlmforum.eu`.

[6] E-ark4all project. `http://e-ark4all.eu`.

[7] Joseph Fong and San Kuen Cheung. Translating relational schema into xml schema definition with data semantic preservation and xsd graph. *Information and software technology*, 47(7):437–462, 2005.

[8] Joseph Fong, Hing Kwok Wong, and Zhu Cheng. Converting relational database into xml documents with dom. *Information and Software Technology*, 45(6):335–355, 2003.

[9] Quang Hoang and Toan Van Nguyen. Extraction of a temporal conceptual model from a relational database. *International Journal of Intelligent Information and Database Systems*, 7(4):340–355, 2013.

[10] Nadira Lammari, Isabelle Comyn-Wattiau, and Jacky Akoka. Extracting generalization hierarchies from relational databases: A reverse engineering approach. *Data & Knowledge Engineering*, 63(2):568–589, 2007.

[11] Mårten Lindstrand. Private Communication, 2019-02-10.

[12] N Mfourga. Extracting entity-relationship schemas from relational databases: a form-driven approach. In *Proceedings of the Fourth Working Conference on Reverse Engineering*, pages 184–193. IEEE, 1997.

[13] Addml (archival data description markup language). https://www.arkivverket.no/forvaltning-og-utvikling/regelverk-og-standarder/andre-arkivstandarder/addml-archival-data-description-markup-language. Accessed: 2018-12-05.

[14] J-M Petit, Farouk Toumani, J-F Boulicaut, and Jacques Kouloumdjian. Towards the reverse engineering of renormalized relational databases. In *Proceedings of the Twelfth International Conference on Data Engineering*, pages 218–227. IEEE, 1996.

[15] Jaana Pinnick. Exploring digital preservation requirements: A case study from the national geoscience data centre (ngdc). *Records Management Journal*, 27(2):175–191, 2017.

[16] Alex H Poole. The conceptual landscape of digital curation. *Journal of Documentation*, 72(5):961–986, 2016.

[17] William J Premerlani and Michael R Blaha. An approach for reverse engineering of relational databases. In *[1993] Proceedings Working Conference on Reverse Engineering*, pages 151–160. IEEE, 1993.

[18] José Carlos Ramalho, Miguel Ferreira, Luís Faria, and Rui Castro. Relational database preservation through xml modelling. In *Proceedings of Extreme Markup Languages 2007 Conference*, 2007.

[19] Ravi Ramdoyal, Anthony Cleve, and Jean-Luc Hainaut. Reverse engineering user interfaces for interactive database conceptual analysis. In *International Conference on Advanced Information Systems Engineering*, pages 332–347. Springer, 2010.

[20] Addml hos riksarkivet. http://xml.ra.se/addml/. Accessed: 2018-12-05.

[21] Om förvaltningsgemensamma specifikationer (fgs). https://riksarkivet.se/fgs-earkiv. Hämtad: 2019-03-13.

[22] Fabio Rinaldi, Oscar Lithgow, Socorro Gama-Castro, Hilda Solano, Alejandra López-Fuentes, Luis José Muñiz Rascado, Cecilia Ishida-Gutiérrez, Carlos-Francisco Méndez-Cruz, and Julio Collado-Vides. Strategies towards digital and semi-automated curation in regulondb. *Database*, 2017, 2017.

[23] Sonja Ristić, Slavica Aleksić, Milan Čeliković, Vladimir Dimitrieski, and Ivan Luković. Database reverse engineering based on meta-models. *Open Computer Science*, 4(3):150–159, 2014.

[24] Siard (software independent archiving of relational databases) version 1.0. https://www.loc.gov/preservation/digital/formats/fdd/fdd000426.shtml. Accessed: 2019-02-13.

[25] Oreste Signore, Mario Loffredo, Mauro Gregori, and Marco Cima. Using procedural patterns in abstracting relational schemata. In *Proceedings 1994 IEEE 3rd Workshop on Program Comprehension-WPC'94*, pages 128–135. IEEE, 1994.

[26] Pedro Sousa, Lurdes Pedro-de Jesus, Gonçalo Pereira, and Fernando Brito e Abreu. Clustering relations into abstract er schemas for database reverse engineering. *Science of Computer Programming*, 45(2-3):137–153, 2002.

[27] Madjid Tavana, Prafulla Joglekar, and Michael A Redmond. An automated entity–relationship clustering algorithm for conceptual database design. *Information Systems*, 32(5):773–792, 2007.

[28] William E Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. *Proceedings of the Section on Survey Research Methods*, page 354–359, 1990.

[29] Dowming Yeh, Yuwen Li, and William Chu. Extracting entity-relationship diagram from a table-based legacy database. *Journal of Systems and Software*, 81(5):764–771, 2008.

Appendix A
Interview Questionnaire

1. Where do you work?
2. What is your field of work?
3. What types of information have you participated in archiving?
4. How long is your experience of archiving information? How long is your experience in databases?
5. How many databases have you archived?
   - 5.1. What type of information did the database contain?
   - 5.2. How large was the database? (Rows or bytes)
   - 5.3. What DBMS was it based on?
   - 5.4. What culling methods have been used?
     - 5.4.1. Are they documented?
     - 5.4.2. Can you share the documentation?
6. What was the most time consuming aspect of the archiving process?
7. Please describe the process you used to archive a database?
8. Did you use any tool in the archiving process?
   - 8.1. Which tool?
   - 8.2. Who have developed the tool?
   - 8.3. What is the purpose of the tool?
   - 8.4. How does the tool improve your process?
9. Have you encountered any especially difficult cases?
10. How do you verify that the extraction of information complies with the archival decision?
11. How do you think the process to cull a relational database for final archiving can be improved?
12. Have you found any patterns in the databases you have archived?

13. If yes, describe the pattern. Do you think it is generally applicable?
14. What information would you as an archivist need to know if a table is required or not?
15. Are there any empty tables preset in your database? How many percent?
16. Have you had any collaborators i the archiving process?
17. Can you recommend any communities for discussing archiving databases?
18. What different fields are present for archiving database?

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden