Christian Markus HOFER, BSc

Data migration to Oracle

Master’s Thesis

to achieve the university degree of

Diplom-Ingenieur

Master’s degree programme: Sofware Engineering and Management

submitted to

Graz University of Technology

Supervisor:

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Nikolai Scerbakov

Institute for Information Systems and Computer Media

Infeldgasse 16c, 8010 Graz

Graz, October 2016

Affidavit

Abstract

The amount of data produced and used by individuals and companies alike is continuously increasing. To handle large amounts of data, different data systems have been developed. For easier data access a piece of software can be built on top of the structured data. Years later it is possible that the used system and its underlying data system will have become outdated because of perpetual technological progress. Negative consequences can be that the system lacks speed or that it might not be possible to represent newly introduced processes on the system anymore. The data or eventually the whole system has to be migrated to a new environment. Data migration can be a risky and highly complex process depending on the systems involved or the data which has to be migrated. To be able to manage the risks of a migration various approaches have been developed. The main focus of this thesis was the migration of various data sources to the Oracle environment. The data sources contain information about samples of tissue which are stored in the Biobank Graz, an institute of the Medical University of Graz. A database was designed and implemented in Oracle 11g. The migration software was implemented in Java ensuring high portability. The theoretical part of this thesis presents and compares data systems, data models and data migration approaches. Additionally, factors that are especially important for the success of the migration are discussed. Finally, the benefits of the developed database compared to the data sources are discussed: The number of redundant data was tremendously reduced, newly introduced constraints help to keep the data stored in the database consistent and the data structure was improved by refining the previous schema(s). The migration algorithm was successfully executed by migrating 100 percent of the consistent data into the new database.

Keywords: Data migration, Oracle, Database Management System, Data Models, Data Systems

Abstract

German

Die von Individuen und Unternehmungen erzeugten und verwendet Datenmengen steigen exponentiell. Softwaresysteme greifen auf Datenverwaltungssysteme, welche aufgrund der stetig anwachsenden Datenmengen notwendig geworden sind, zu. Diese Systeme haben ein Ablaufdatum und laufen aufgrund sich beständig verändernder Anforderungen Gefahr antiquiert zu werden. Beispiele für wechselnde Anforderungen die eine Migration notwendig machen sind neue eingeführte Geschäftsprozesse, welche vom bisherigen System nicht mehr abgebildet werden können, Wechsel auf eine nicht unterstütze Soft- oder Hardwareplattform, Verlangsamung des bestehenden Datenverwaltungssystems aufgrund anwachsender Datenmengen, etc. In solchen Fällen müssen die Daten bzw. das Softwaresystem auf eine neue Plattform migriert werden. Migrationen sind oftmals risikoreich und von hoher Komplexität geprägt. Um die Risiken von komplexen Migrationen eindämmen zu können wurden verschieden Konzepte entwickelt. Das Hauptziel dieser Diplomarbeit bestand darin, eine Datenmigration von verschiedenen Datenquellen (MySQL, Microsoft Access, Oracle, XML) auf eine neu designte und implementierte Datenbank in Oracle 11g durchzuführen. Die zu migrierenden Daten beinhalten Informationen zu Gewebeproben, welche in der Biobank Graz, ein Institut der Medizinischen Universität Graz, asserviert werden. Aus Gründen der Portabilität wurde die Migration in Java implementiert. Der theoretische Teil der Arbeit beschreibt und vergleicht Datenverwaltungssysteme, Datenmodelle und Konzepte zur Datenmigration. Zudem werden die Erfolgsfaktoren der Datenmigration näher beleuchtet. Im letzten Kapitel wird die neue Oracle Datenbank den Quelldatenbanken gegenübergestellt. Das Resultat ist eine besser strukturierte Datenbank, bei welcher die Anzahl der redundanten Inhalte beträchtlich verringert werden konnte. Zusätzlich verhelfen neu eingeführte Restriktionen in der Zieldatenbank zu konsistenten Datensätzen. Im Zuge der Migration konnten sämtliche, als konsistent detektierte Datensätze in das neue System erfolgreich migriert werden.

Stichwörter: Datenmigration, Oracle, Datenbankverwaltungssystem, Datenmodell, Datenverwaltungssysteme

Acknowledgement

I would first like to thank my thesis advisor Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Nikolai Scherbakov, whose office was always open whenever I had a question about my research or writing. He was able to steer me in the right direction whenever I did not know exactly where to go next.

I would also like to thank Vr.-Ped. Karine Sargsyan who made this thesis possible. Additionally, I would also like to thank Robert Reihs of the BBMRI and all the colleagues of the Biobank of Graz, especially the following experts: Ing. Robert Primschitz, Dr. Franziska Vogl, Dr. Sebastian Fuchs and Dr. Tanja Macheiner who helped me to get wider and deeper knowledge regarding this thesis.

I would also like to acknowledge Dipl.- Ing. Robert Ladstätter for his suggestions and advices, which were greatly appreciated. Additionally, I would like to thank my brother, Andreas Arslan, BSc. for giving me advice in writing this thesis.

Finally, I must express my very profound gratitude to my parents and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Table of contents

List of figures

List of tables

Listings

Glossary

ANO	Admission Number
ANSI	American National Standards Institute
API	Application Program Interface
BBMRI	Biobanking and BioMolecular resources Research Infrastructure
CODASYL	Committee on Data Systems Languages
CPU	Central Processing Unit
CSV	Comma-Separated Values
DB	Database
DB4O	Database For Objects
DBMS	Database Management System
DBS	Database System
DM	Data Model
DVCS	Distributed Version Control System
ERM	Entity Relationship Model
ETL	Extract, Transform, Load
FFPE	Formalin-Fixed Paraffin-Embedded
FOS	File-Oriented System
GUI	Graphical User Interface
HCI	Histologic Circle
HER2	Human Epidermal growth factor Receptor 2
HDM	Hierarchical Data Model
HNO	Histologic Number
HTML	Hyper Text Markup Language
ICD	International Classification of Diseases
ID	Identifier
IDE	Integrated Development Environment
IDS	Integrated Data Store
IMI	Institute for Medical Informatics, Statistics and Documention
IMS	Information Management System
IT	Information Technology
JDBC	Java Database Connectivity
JVM	Java Virtual Machine

JRE	Java Runtime Environment
LN2	Liquid Nitrogen
MS	Microsoft
MySQL	My Structured Query Language
NDM	Network Data Model
ODBC	Open Database Connectivity
ODM	Object Data Model
OO	Object-Oriented
OS	Operating System
ORM	Object Relational Model
PHP	PHP: Hypertext Preprocessor
PL	Procedural Language
PYPL	PopularitY of Programming Language
RAM	Random Access Memory
RDBMS	Relational Database Management System
RDM	Relational Data Model
REGEX	Regular Expression
SGML	Standard Generalized Markup Language
SQL	Structured Query Language
XML	Extensible Markup Language

1 Introduction

The amount of data is continuously increasing. This applies to data generated by individual human beings as well as to companies. To remain competitive, the efficient management of data and the data itself have become critical objectives of an organisation. Databases (DB) help people to manage data efficiently and allow an extraction of the right information at the right time thus potentially supporting the success of an organisation (1,2).

DBs are integrated in our daily life: It is likely that a DB is accessed when purchasing goods at the supermarket, using a credit card, booking holidays at the travel agency, renting a book at the library, etc. (2).

1.1 Data systems

1.1.1 File Oriented System (FOS)

The successor of labelled folders, which were stored in cabinets eventually locked for security reasons and only made accessible to employees entrusted with the cabinets key was the FOS. Formerly instead of DBs FOS were often used in organisations: The conventional FOS is decentralized: Each application in the information system has its own files containing data which could be stored in another file used by a different application. These FOSs with their unrelated files became largely obsolete because of the continuously increasing complexity of business. FOS come inter alia with the following disadvantages:

- Data redundancy
- Data inconsistency
- Program-data dependence
- Poor data control
- Limited data sharing
- Inadequate data manipulation capabilities

A redundancy of data is the result of decentralisation by using the same e.g. Identifiers (ID) in different files which are accessed by independent application programs. Data redundancy implies a loss of data integrity or an inconsistence within the data because there may be discrepancies between data formats and data values. Program-data dependence means, that each program using data files needs a file description - respectively metadata - of the accessed files. A file description contains inter alia the physical structure and the storage of a file. The decentralisation of FOSs results in poor data control. One distinct field with a specific meaning can be used in various files and it is possible that the identifier of the field differs in each file. Such a case could lead to different meanings because of the different field names – respectively identifiers. In special cases the same identifiers do not have the same meaning which leads to poor data control. Data sharing is limited because each application uses its own private files, which are not shared. The separate files of a FOS are not connected to each other and this results in a limitation of data manipulation capability (1,3).

1.1.2 Database System (DBS)

DBS eliminate the issues of FOSs, see section 1.1.1. They base on a centralised and integrated data structure so that problems like data redundancy and poor data control are prevented. DBSs comprise related data, which is logically structured and shared. DBs are embedded in database management systems (DBMS), which are essential for various types of organisations such as business companies, banks, universities, etc. DBMSs manage the users' data access to the physical DB. Such a system leads to several advantages such as:

- Minimal data redundancy
- Improved data consistency
- Program-data independence
- Improved data sharing
- Improved data integrity
- Increased concurrency

Data redundancy in DBS can easily be avoided because of the centralized control of data. Redundant data, which can be useful in some cases can be controlled in a consistent manner. As mentioned in section 1.1.1, data redundancy results in inconsistency. DBS minimalize data redundancy and help to achieve consistency. Program-data independence is guaranteed by separating the metadata from the application accessing the data. The metadata is stored in the DBS. The data within the DB can be shared by all application accessing the centralized repository. Users and applications can only access data if they are authorized to do so. Consistent and accurate data lead to a high level of data integrity. The integrity of a DB can be increased by adding constraints to the DBS. Constraints are special rules for the data, which are stored in the DB. Multiple users can access DBMSs, because these systems eliminate the issues of concurrency. (1,3).

1.1.3 Overview and comparison

A general overview of the properties of the different data systems is shown in table 1.1. FOSs are organized as file records and DBMSs comprise related tables. The access to multiple tables at a time is made possible by the use of a DB system. The FOS on the contrary allows the access to a single file at a time. The FOS coordinates the access to physical data. In contrast to the FOS the DBMS additionally manages the logical access to the data. Data redundancy of the FOS can be very high which causes a higher occupation of storage and eventually inconsistences within the stored data. The redundancy is often caused by different departments using their own private files containing eventually duplicated data. DBMSs have a centralized DB which makes it possible to minimize a redundancy of data. Concurrency issues have to be handled explicitly in the FOS. The DBMS is designed to grant data access to multiple users simultaneously. Indices and unique keys, which help inter alia the user to find data more rapidly can be used in the DBMS but not in the FOS (1,3).

	FOS	**DBMS**
Organization	Files	Tables
Access	Single file at a time	Multiple tables at a time
Data access	Physical	Physical and logical
Redundancy	High	Low
Concurrency	More restrictive	Handled by the DBMS
Indices or keys	No	Yes
Location	Decentralized	Centralized

Table 1.1 Comparison of properties between the file system and the database system (1,3)

1.2 Data Models (DM)

The DB model aka DM describes the logical structure of a DB and the operations on a DB. The structure includes data types, relationships and constraints. Operations allow to access the data, which entails retrieval, modification, deletion, insertion, etc. In this section the following DMs are presented: hierarchical, network, relational, object oriented and the object relational model (3–5).

1.2.1 Hierarchical Data Model (HDM)

Lots of computer users are familiar with a hierarchical organized data structure: The file system of an operating system is very often structured in that way. Files and folders are used to order data in such a way that it can be found later. As these file systems the HDM structures data by using a tree data structure, which means that each record belongs to one owner. The central characteristics of such a model is that it can contain one to one and many to one relationships between entities. A child entity can be only the child of one entity. Figure 1.1 shows an example of a hierarchical DM (3,4).

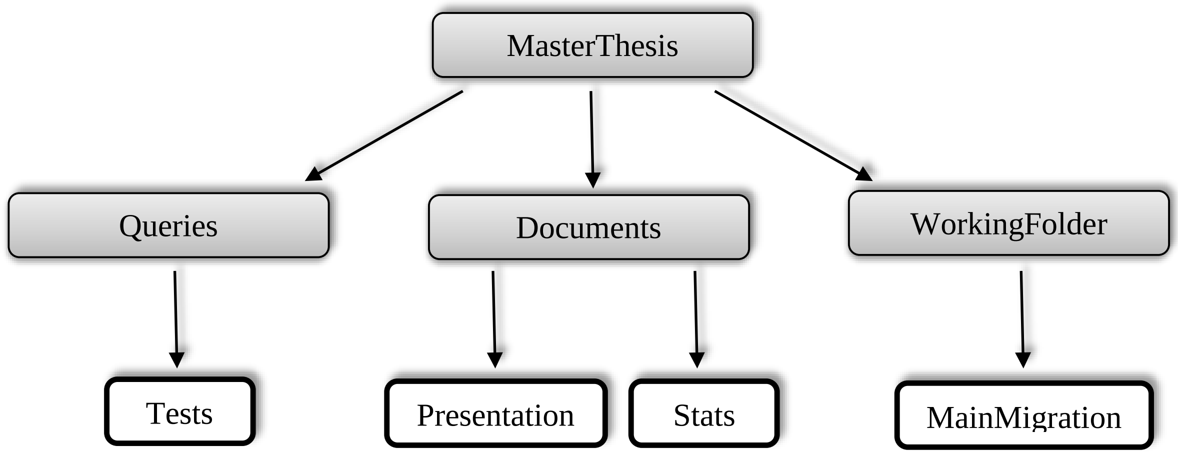


Figure 1.1 Example of a hierarchical data model: The root entity is "MasterThesis", which represents the top of the hierarchy and comprises the children "Queries", "Documents" and "WorkingFolder". The node "Documents" comprises the records "Presentation" and "Stats". These records are owned by "Documents" and it is not possible that these two records emerge in another node (4).

The HDM is often described as a navigational DM: To find a specific record the hierarchy has to be went through until the record is found. An example for a very well-known hierarchical database model is the *information management system* (IMS) of *IBM*[1], which is used since 1960 and still supported by IBM (4,5).

1.2.2 Network Data Model (NDM)

The great difference between the NDM to the HDM, see section 1.2.1 is the ability to use many to many relationships between entities. This implies that in the NDM a child entity can be the

[1] http://www.ibm.com

child of multiple entities. The addition of the many to many relationship allowes a more realistic abstraction of many scenarios. Figure 1.2 illustrates a network model using the possible relationships (3,4).

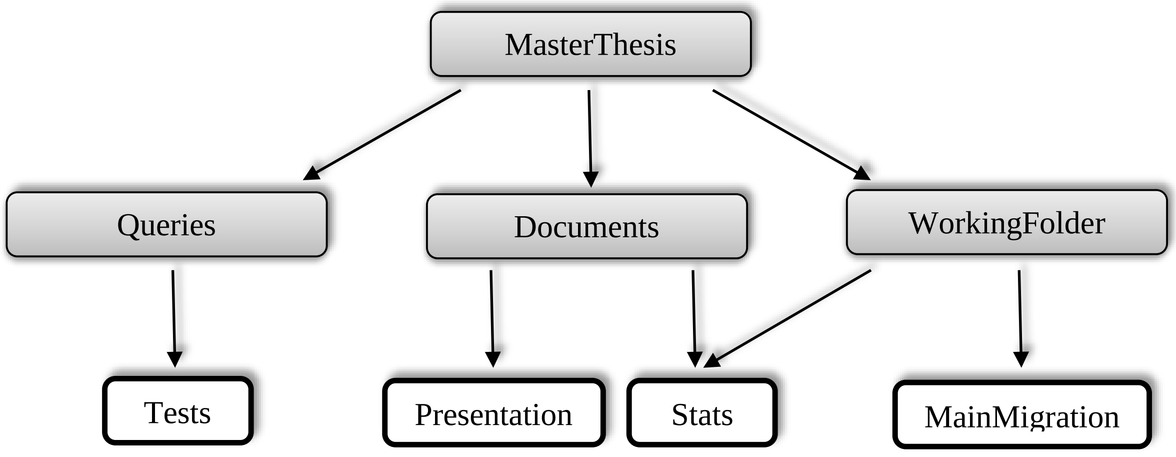


Figure 1.2 Example of a network data model: The root entity is "MasterThesis", which represents the top of the hierarchy and comprises the children "Queries", "Documents" and "WorkingFolder". The node "Documents" comprises the records "Presentation" and "Stats". The "WorkingFolder" comprises the record "MainMigration" and the "Stats", which is a child of the node "Documents" too. In the network model it is possible, that one child has one or more parents (4).

Like the HDM, see section 1.2.1 the network model is a navigational model that provides more flexibility because of the added many to many relationship in comparison to the hierarchical model. The NDM was introduced by the Committee on Data Systems Languages (CODASYL) in the 1960s. An example of NDM is the *Integrated Data Store* (IDS) of *Honeywell*[1] (4,5).

1.2.3 Relational Data Model (RDM)

The RDM is different to the models discussed previously. This model represents data entities and relationships by the use of tables. Each table structure has a number of columns, which contain unique names or values, called primary and foreign keys (3,4).

[1] http://www.honeywell.com

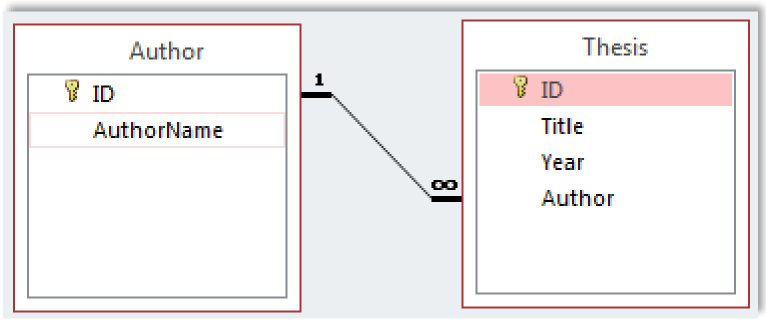


Figure 1.3 Example of a relational model: The two tables "Author" and "Thesis" are linked together by the use of a one to many relationship. Each "Thesis" and "Author" has a unique identifier, which is identified by the icon of the yellow key next to the ID field in each of the tables. The relation defines that a thesis is written by one author and one author can write many theses.

To minimize redundancy RDM design is based on normalization. The RDM emerged in the 1970s and since then it has been the most widely used data model. Figure 1.3 illustrates an example of a relational model (3,4).

1.2.4 Object Data Model (ODM)

The object-oriented (OO) paradigm is based on objects. Each object comprises a number of variables holding values of the object. Each object comes with its own methods, which are procedures operating on the object. Objects with the same variables and methods are instantiated from the same class. In OO programming languages the object's lifetime can span from the start until the termination of an application. The ODM on the other hand does not store the objects in the application memory, which makes them available after the termination of the program. Language bindings allow the programmer to use and manipulate the persistent objects. The access to the objects is defined in the design model. Like the hierarchical model, see section 1.2.1 and the NDM, see section 1.2.2 the ODM is navigational. It was introduced in 1989. An example for an ODM is *database for objects*[1] (DB4O) (3,4).

1.2.5 Object Relational Model (ORM)

The motivation behind the development of the ORM was to keep the advantages of the RDM and extend it by making it possible to store more complex entities like objects in the DB. The ORM can store objects with their attributes and methods in fields or in relational tables. This

[1] http://sourceforge.net/projects/db4o

kind of DM emerged in 1990 and was introduced by Michael Stonebraker. The ORM is supported by various DBMS like *Oracle*[1], *Microsoft (MS) Corporations SQL server*[2] and *IBM's DB2 server*[3] (3,4).

1.2.6 Overview and comparison

In the 1960s the first DB models were developed. Before that time, applications accessed files to retrieve or manipulate data, which caused lots of barriers, see section 1.1.1. In the first generation of DMs the hierarchical model and the network model were developed. These kinds of DBs have mostly disappeared by now but there are still companies offering such DBs and supporting their vendors. In 1970 the second generation of DMs emerged consisting of the widespread relational DM, which was proposed by Edgar Codd. The third generation of DMs comprises models, which deal with the problems arising when relation DBs meet object-oriented systems in the 1980s. The post-relational models comprise inter alia the ODM and the ORM. Table 1.2 shows a comparison of the mentioned DMs in section 1.2 (3–5).

	HDM	**NDM**	**RDM**	**ODM**	**ORM**
Data element organization	Files, records	Files, records	Tables	Objects	Objects
Relationship representation	Tree	Graph	Foreign key concept	Logical Containment	Relational extenders
Access language	Procedural	Procedural	Non-procedural	Procedural	Non-procedural
Introduced	1960s	1960s	1970	1989	1990

Table 1.2 Comparison of data models: The HDM and the NDM are organized in records. The RDM uses tables and object based Models handle with objects. Relationships between entities are represented in the HDM by trees, in the NDM by graphs, the RDM uses the foreign keys, the ODM uses encapsulation of the OO paradigm and the ORM extends the RDM by the use of objects. RM and the ORM are accessed by non-procedural languages. The rest instead is accessed by procedural languages (3,4).

1 https://www.oracle.com
2 https://www.microsoft.com/en-us/server-cloud/products/sql-server
3 http://www-01.ibm.com/software/data/db2

1.3 Database Management Systems (DBMS) popularity

In December 2015 the three most popular DBMS ranked by the popularity: *Oracle*, *MySQL* and *Microsoft SQL Server* (MSSQL). Oracle and MySQL play an important role regarding the practical aspects of this of this thesis. Additionally *Microsoft Access* (MSA) is being as well because of its significance in the practical part (6).

Rank	**DBMS**	**Score**
1.	Oracle	1497.55
2.	MySQL	1298.54
3.	MSSQL	1123.16
4.	MongoDB	301.30
5.	PostqreSQL	280.09
7.	MSA	140.21

Table 1.3 Popularity ranking of DBMS: The table gives an overview of the five most popular DBMS systems y ranked by DB-Engines. The ranking illustrates the dominance of the RDM. The first non-relational DB is MongoDB ranked on the 4th place. MSA is ranked on the 7th place (7).

The ranking, see table 1.3, is from the end of December 2015 and was calculated by *DB-Engines*[1]. Figure 1.4 shows the popularity trends of the four DBMS, which are discussed in this section. Basically it can be said that the popularity score of the mentioned relational DBs has stagnated. Additionally, the trend of *MongoDB* was added, whose popularity score has significantly increased during the last two years. MongoDB is a document based DBMS. The score is influenced by different parameters:

- The number of mentions on the Internet, which is calculated by the results of *google*[2] and *bing*[3].
- The interest in the system, which is measured by the search frequency according to *google trends*[4].
- The number of interested users on *Stack Overflow*[5] and *DBA Stack Exchange*[6].

[1] http://db-engines.com
[2] https://www.google.com
[3] http://www.bing.com
[4] https://www.google.com/trends
[5] http://stackoverflow.com
[6] http://dba.stackexchange.com

- The number of job offers on *Indeed*[1] and *Simply Hired*[2].
- The number of professional networks using the system mentioned on *LinkedIn*[3].
- The number of *twitter*[4] tweets mentioning a DB system (6).

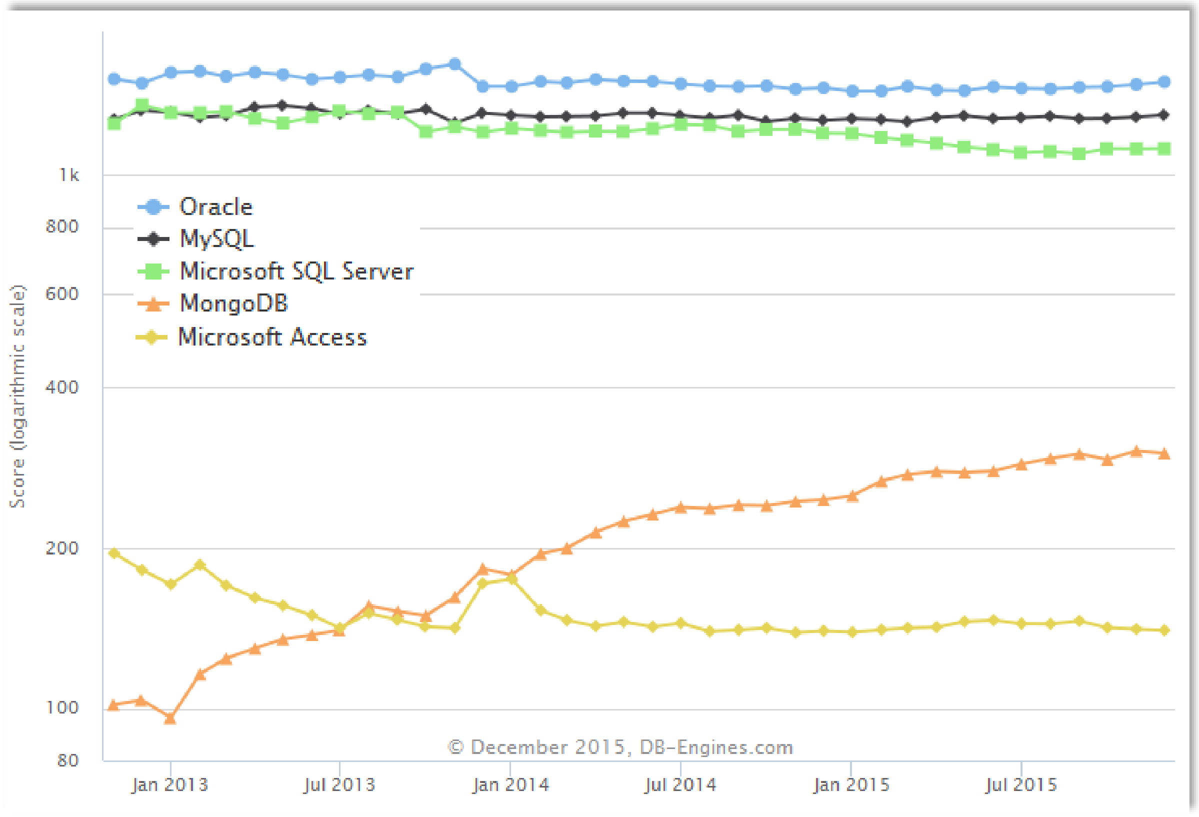


Figure 1.4 Popularity ranking trend of the compared DBMS in this section from November 2012 to December 2015. The y axis shows the score logarithmically and the x axes the month and year. Oracle is the most popular DBMS followed by MySQL and MSSQL who switched their position various times especially in the time from November 2012 and September 2013. After a big gap follows the first non-relational DB called MongoDB, which almost tripled its popularity score in the last two years. MSA seems to be a popular DB too but has plateaued for 2 year now at a score of about 130 (8).

DB-Engines is an initiative collecting and viewing information about DBMS. Beside established relational DBs NoSQL DBs like MongoDB are considered. The site includes a database encyclopedia offering explanations of concepts and terms and it is possible to view and compare important properties of various DBMS (6).

1.4 Data migration

[1] http://at.indeed.com
[2] http://www.simplyhired.com
[3] https://www.linkedin.com
[4] https://twitter.com

Data migration is the permanent movement of data from old systems (computers, storage devices, formats, etc.) to a new repository. A migration may be performed for several reasons e.g. for website consolidation, data center relocation or storage equipment replacements or upgrades, etc. The migration data has to be

- Selected,
- Prepared,
- extracted and
- transformed.

A selection of data for example has to be performed if multiple sources of data exist. Preparation, extraction and transformation are important to ensure data quality. A data migration for enterprise applications is generally a permanent movement from the source to the target. Besides the mentioned activities the legacy data has to be decommissioned. Data migration can be categorized into database (DB) migration, storage migration, application migration and business process migration, see section 1.5 (9–11).

1.5 Data migration categories

1.5.1 Database migration

The migration of data between different computing platforms increases the flexibility of enterprise information business operations significantly. Even though the advantage is a tremendous one, such a movement of data represents often a great challenge because of its complexity: Incompatible file system metadata formats, volume metadata formats and data formats in application files. That is the reason why Information Technology (IT) departments often do not migrate and run their system on less than optimal platforms. This results in a disadvantage and datasets become captive to the server platform that processes it (12).

The term DB migration is a broad one and the complexity may vary greatly. The easiest form of DB migration would be the movement of data from one DB to another, where all the extracted data from the source DB are migrated to the new one. The term may also refer to a migration between two database management systems (DBMS) (11).

1.5.2 Application migration

Application migration refers to the movement of an application from one environment to another. Such a migration is necessary when a company for instance decides to change their software. A concrete example is the migration from an On-premises Software to a cloud provider's environment. Data from the source application is extracted, transformed and loaded into the target application. This kind of migration can be complicated because the mode of operation may differ enormously between the two applications (11,13).

1.5.3 Business process migration

Data is often strictly aligned to business processes of an organization. Hence, a change of one or more business processes often results in a change of software. Business process migration is very similar to application migration but includes additionally the change of a business process, which makes the migration a bit more complicated (11).

1.5.4 Storage migration

The term storage migration simply refers to the movement from an outdated storage location to a new place for data storage. In general there are no essential data changes introduced (11).

1.6 Data migration approaches

1.6.1 "Classic" data migration model

The classic data migration model, see figure 1.5 comprises five processes:

1. Milestone: Project Start
2. Process: Design
3. Process: Extract, transform, load
4. Process: Test
5. Milestone: Project Sign-off

Typically, business briefs the IT team broadly about the upcoming data migration, focusing on a safe migration without disturbing business. The goal of the design process is the preparation of the data for movement by specifying e.g. the mapping or movement rules. After the design process the source data is extracted, transformed and loaded into the target system. The testing process follows after the migration. If testing fails, the process starts all over again from the design phase. After successful testing the project is ready to be signed off (14).

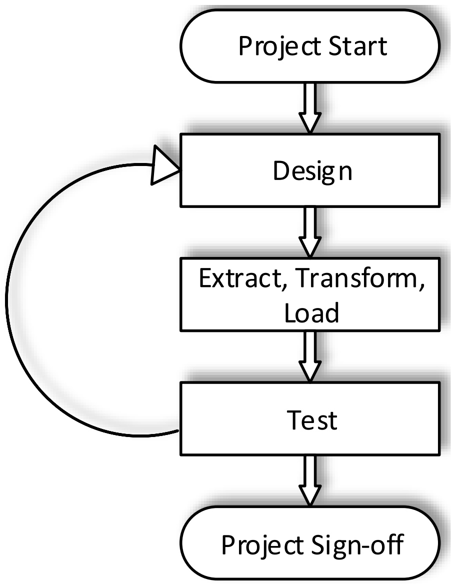


Figure 1.5 Classic data migration process: Usually the project starts with a broad brief including the data source(s) and the target system from the management to the IT team. The design process follows after the project start. In this process inter alia ad-hoc and sample data analysis are made. After the design process data is extracted from the source, transformed and loaded into the target system. Finally, the migrated data is tested. Often testing reveals errors. A new iteration starts from the design process. This procedure is repeated until the test is successful. After the accomplished tests the project is ready to be signed off (14).

1.6.2 Revised data migration model

An effective data migration project, see figure 1.6, starts with the planning phase. In this stage the goal of the migration is defined. Data sources are analyzed and the scope of data, which has to be migrated is refined. Analysis should be performed in cooperation with the business users who work with the target system in future. Data refinement is an integral part of a successful migration. Additionally, the planning phase comprises the timeline, the resource and budget plan. The second phase is called profile & audit. In this stage the content of the sources is identified. Profiling and auditing tools help to understand the data sources. By understanding the data sources, it can be decided which sources will actually be migrated. Additionally, conflicts and inconsistences are identified and viewed in detail to be able to solve them. By applying this approach, it is possible to identify anomalies soon. Issues identified that way influence the next phase of the migration - the design phase: In this stage the mapping specifications are made. Rules are defined, which are based on the analysis made in the previous phase. The rules have to be fully understood by the business and signed off. After that, the migration code is implemented and is matched to the rules specified to identify possible errors (14).

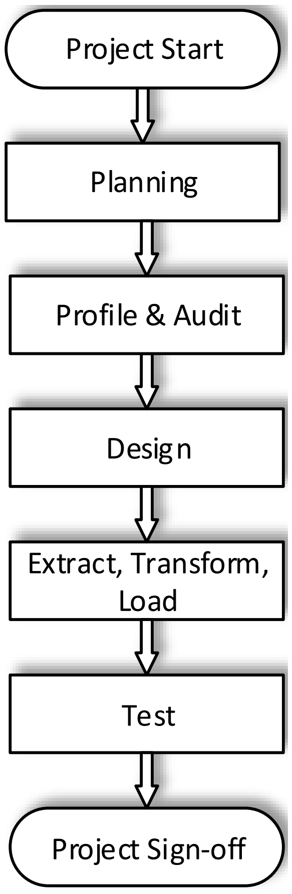


Figure 1.6 Revised data migration model: The first phase after the start of the project is the planning phase. The goal, timeline, budget and resouce plan is defined. In the profile and audit phase the actual data which is migrated is defined and inconsistencies and conflicts are identified. With the knowlegde gained in phase two the mapping rules are defined and implemented in the design phase. The design phase is followed by the execution phase where the source data is extracted, transformed and loaded into the new system. After migration, unit, system, volume, online application and batch application tests are executed. The migration project can be signed off after succesful testing (14).

Phase four deals with the execution of the migration. The source data is extracted from the source system, cleansed, transformed and loaded into the new system. The goal of the final process is the testing of the migrated data. Returning to the design phase should not be necessary, unless relevant changes have taken place. In this case the profile and audit phase has to be extended. After testing, the migration project is ready to be signed off by the business (14).

1.6.3 Data migration process model

Data migration is a onetime process, which is influenced by many different factors.

Figure 1.7: Data migration process model, which consists of four stages: Initialization, Migration development, Testing and the Cut-over. Each of the stages contains several phases. Each phase yields a result called deliverable (15).

illustrates the data migration process model for large business projects which was publicized in 2012. The process model is divided up into four stages: Initialization, migration development, testing and cut-over. Each stage consists of several phases. Each phase yields a result called deliverable. In the stage of initialization, the project organization and the technical infrastructure are set up. The responsibilities and the migration strategy are defined. Additionally, the migration platform is set up in the first stage. The second stage deals with the development of the migration. The source and the target structure are analyzed and a mapping is defined. After precise planning, the source data is unloaded by the use of unload scripts. Additionally, the source data has to be cleansed. Transformation rules are defined and implemented. This step is necessary because usually the source and the target data models are different. The result of stage two is the transformed data. Several iterations within this stage are possible until the migration has been completed.

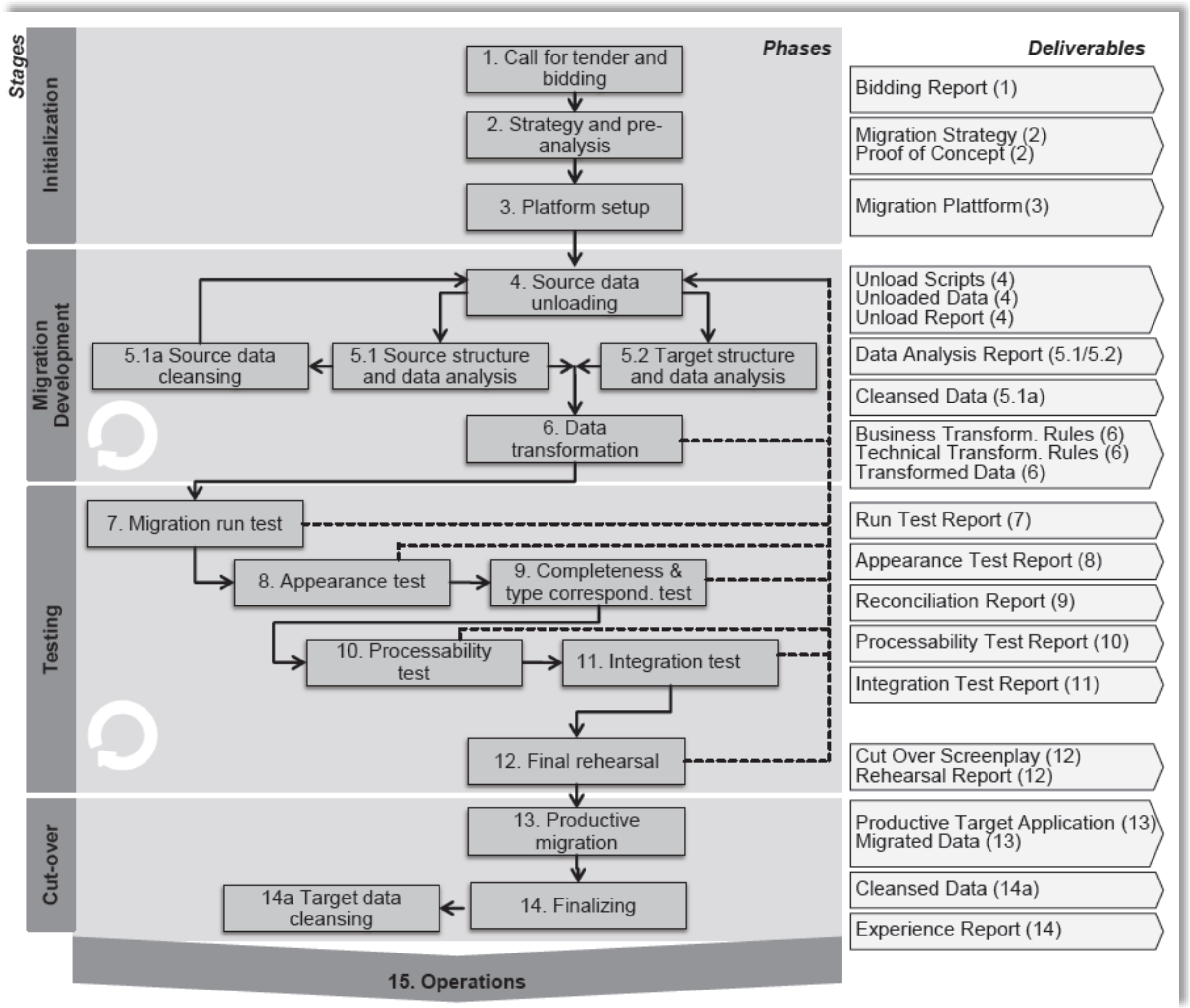


Figure 1.7: Data migration process model, which consists of four stages: Initialization, Migration development, Testing and the Cut-over. Each of the stages contains several phases. Each phase yields a result called deliverable (15).

The focus of stage three is the testing of the migrated data. Before testing, the infrastructure has to work correctly. The migration run test is executed and the result is the run test report, which creates a log file where the errors and the execution time are recorded. The appearance test report clears up if the source and the target data are semantically equivalent after migration. The processability test ensures that the processes were executed correctly. The integration test is executed if the application is embedded in a network of other applications which are connected together. The associated test report gives information about the errors and passed processes concerning the application network. Analogous to stage two, several iterations may be necessary to finish the testing stage. The final stage deals with the integration of the system into the operative business. The first phase in this stage is the productive migration. The old system is shut down after the source has been migrated to the target. The phase of finalizing starts after initialization of the target system. If everything is fine the migration is finished. An experience report is made to give an overview of the migration and includes important information which can be used for future migration projects. If it was not possible to cleanse the source data completely an additional cleansing of the target data may be necessary after the finalizing phase (15).

1.6.4 Challenges and risks of data migration

- Little input by the business and much input of the IT department can lead to a movement of unnecessary data, eventually causing more costs and longer migration times.
- If only small sets of data of the source system are analyzed, it is possible that assumptions are made that cannot be upheld when taking the entire dataset into consideration. This leads to errors and a high rework rate (E.g. testing failures causing more iterations and cause an additional loss of time).
- Frequently, mapping specifications are metadata driven and not content driven. Because the movement of data is based on analysis of small data samples and ad hoc queries of the source system, little knowledge of how the source or target system works (14).

1.7 Data migration strategies

1.7.1 Big Bang data migration

This strategy has existed for decades now and is an uncompromising one. Data migrations which move an entire dataset in one operation are following the so called Big Bang migration strategy. Before the actual process the whole system is prepared for the data migration by shutting down the applications and DBs. Work is stopped and everything is focused on the migration. Migrating in such a way generally guarantees a short migration time. Big Bang migrations in companies are often executed after work hours or over the weekend. E.g. on Friday users work on the legacy system and on the following Monday they are using the new target system. Sometimes there is a big switch and the users are confronted with new operating procedures or frontends, but it is possible that they do not recognize the switch because no changes are made on the frontend (16,17).

A Big Bang migration is often a great change. This migration strategy may be risky and that is the reason why a lot of companies decide to run the legacy and the target system in parallel. If these systems are running in tandem, they have to be synchronized, see section 0. The technological progress makes such a synchronization easier. If errors occur after a big bang migration a backspace to the legacy system would be possible at any time. In case of passing the functionality and data tests the company can shut down the legacy system (16,17).

1.7.2 Iterative data migration

This migration strategy is also known as phased data migration, trickle feed data migration or synchronized data migration. The core of this migration strategy is the movement of data in small pieces until the whole data is moved from the legacy to the target system. In contrast to the previous mentioned approach is that specific parts of the system are shut down and not the whole system. Additionally, there are always two systems running in parallel until the migration has succeeded. This strategy is an interesting approach for companies which are running twenty-four hours a day (16,17).

The process of iterative migration has to face two main challenges:

- Synchronization between the legacy and the target system
- Migration of specific parts and functionality without disturbing the overall business continuity

Concerning the issue of synchronization, it has to be considered whether the migration includes an entirely new application or the old application with moved data. If the migration includes a new application, business operations are running across the legacy and the new system. In this case both systems need a data synchronization. Such synchronization can be mono-directional or bi-directional, see section 0. If the application is not changed and an iterative DB migration is performed the program needs to know where the parts of the data source have moved (16).

Migrating the business functionality iteratively should be carried out by using a pragmatic approach: Temporary methods or procedures have to be developed to handle the split up (16).

"If you have a nationwide parts system you could migrate iteratively by a particular manufacturer e.g. Ford or BMW. When customers ring up for a specific manufacturer your users can be trained to use the appropriate system (16).*"*

The iterative data migration is not as frequently used as the Big Bang data migration (16).

1.7.3 Comparison

In this section the different migration strategies mentioned in section 1.7 are compared.

	Big Bang	**Iterative**
Time of migration	Short	long
Obligatory system downtime	Yes	No
Riskiness	High	Low
Synchronization needed	No	Yes
Migration process size	Large	Small steps
Number of migration processes	One	More

Table 1.4: Overview and characteristics of data migration strategies (16,17)

Table 1.4 illustrates the differences between the Big Bang and the Iterative data migration strategy. Usually the migration time is shorter when using the Big Bang migration because this strategy contains only one big migration process. The iterative approach consists of a number of smaller migration steps, which are made over time, where specific parts of the source are migrated. Thus, the iterative method is not as risky as the Big Bang approach, because if one small step fails it is easier to roll back than one huge migration process. The system has to be completely shut down when using Big Bang. By applying iterative migration this can be avoided (16,17).

1.8 Synchronization methods

In some cases, especially when an iterative data migration is performed the source and the target system have to be synchronized to receive the current data. Basically synchronization can be divided up into mono-directional and bi-directional synchronization (16–18).

1.8.1 Mono-Directional synchronization

New data is stored either in the legacy or in the target system. Changes being made in one of the systems are being synchronized with corresponding data in the other system. E.g. the legacy system is synchronized with the target system receiving the changes. If issues emerge within the target system business services can be redirected to the legacy system (18).

1.8.2 Bi-Directional synchronization

Both systems, the source and the target, automatically synchronize if changes are made in either one of them. Typically such a synchronization method is used if an iterative data migration with parallel running takes place (18).

1.9 Database migration

1.9.1 Migration of databases between different RDBMS

The migration of DBs between different types of relational database management systems (RDBMS) is generally processed in three phases: extraction, transformation and loading. In the extraction phase data is extracted from the source DB. During the transformation step the data is transformed in a way that allows it to be migrated to the source DB. In the third phase the data is migrated to the target DB. One of the main differences between the RDBMS are the naming and the support of various data types, e.g. the data type *varchar* does not exist in Microsoft (MS) Access but the system contains a type, called *text* which can be used that way (19,20).

There exist several tools for migration between two RDBMS like *DatabaseBridge* or *Data Management Center*. Using such tools is a very convenient way to migrate DBs. Some of them offer the possibility to compare the source and the target tables of the migrated DB. The other side of the coin is that these tools are limited in functionality and some of them can cause issues e.g.: Not all of them are able to migrate the foreign keys contained within the tables. It is not possible to change parameters or to include additional features of the target DB tables and their attributes like data types. Also these migration tools are very inflexible because they do not offer the possibility to add extensions manually (19).

1.10Legacy information system migration approaches

Computer systems and computer technologies have been used for decades. Over this time their complexity has increased enormously and some systems became outdated but are still running, because of the fact that without these outdated running systems the business would halt. Thus, these systems have to be maintained. These so called Legacy information systems resist further modification and revolution and present a crucial number of problems, e.g.

- Maintenance costs are generally high because of the use of obsolete hardware. Old hardware is typically slower than modern ones and this reduces productivity.
- In general, high maintenance costs occur by maintaining old software products. Documentation is missing and the knowledge could have been lost which causes time consuming maintenance, which results in higher costs.
- It is very difficult to add functionalities to such systems, which causes a disadvantage for the organization using it.

Data which is stored in these legacy information systems are resources representing important knowledge. To be able to set up a new system data held in the legacy system has to be migrated. This takes a lot of effort and it is possible that the approach chosen is not a successful one. That is why it is possible that organizations avoid the migration of legacy systems. The legacy system can become so essential for important processes that I cannot be easily removed, thus blocking progresses. Because of this methods have to be provided to migrate old systems to new platforms and architectures (21,22).

1.10.1 Database First migration

The Database First migration is also known as Forward Migration. Initially, as the name suggests, the data is migrated first into a modern data system. After the data migration the legacy applications, interfaces and functions are incrementally redeveloped. During this redeveloping the legacy system is still running. The outdated system can access the new DB by using so called Forward Gateways. The gateway translates the requests of the legacy system and redirects the call to the new DB service system. The results are returned to the gateway, which translates it back so that the source system may operate correctly (21).

1.10.2 Database Last migration

The Database Last method is also known as Reverse Migration. In contrast to Forward Migration, the data remains on the original platform until the target system is redeveloped. The final step of DB last migration is the movement of the data to the new system. During the system migration phase a Reverse Gateway allows the target system to communicate with the legacy data management service. As in section 1.10.1, the gateway is used for translation and redirecting calls and results (21).

1.10.3 Cold Turkey migration

Applying the Cold Turkey strategy means a redevelopment of the legacy system by applying modern software techniques. Modern DBs, execution platforms and software architectures are used. In general, the functionalities of the legacy system and additional features are included in the target system. Such a project takes usually much time because development is started from scratch. Due to the long development time and the high complexity of such a migration, this method is risky e.g. during development new technologies may emerge and business processes may change. A worst case scenario could be the use of an outdated technology but nevertheless Cold Turkey could be the appropriate approach if well-defined and stable legacy systems are migrated (23,24).

1.10.4 Composite Database migration

In this migration approach the target and the legacy system are running in parallel. During the migration process the target system is growing in size and at the end of the process it covers the whole functionality of the source. Forward and backward gateways are used for communication between the two systems during the migration. To achieve integrity of data, a transaction coordinator is installed. Composite Database migration is performable on fully decomposable, semi decomposable and non-decomposable legacy systems (24).

1.10.5 Chicken Little migration

The legacy system is migrated to the target system by the use of small incremental steps. Each step covers specific results and approximates the system to the overall goal, the complete coverage of the legacy system. The Chicken Little approach refines the Composite Database method introducing different types of gateways: Forward or backward (DB) gateways, application gateways and information system gateways are used. DB gateways are used to grant access to the DB service, application gateways are located between the user and system interfaces of the legacy system. Information system gateways are located between users, external information systems and the legacy system (24).

1.10.6 Butterfly migration

The core of the butterfly migration approach is data migration. However, the general task of this approach is legacy system migration. The goal of this method is to provide a safe, fast and simple migration of a mission-critical legacy system. During the migration the complete live data is stored in the source system. The target system is initialized after the finalization of the migration. This means that the butterfly approach does not use gateways, which are used to distribute data to the source and target system. The great issue of data consistency is bypassed. The shutdown time of the legacy system is very short in comparison to other approaches like Big-Bang migration. The migration process comprises six phases:

- In ***phase 0*** preparations for the migration are made. The definition of the user requirements and the determination of the target system are the most important parts of this phase.
- **Phase 1** focuses on the understanding of the legacy system and the development of the future data schema(s).
- The main parts of **phase 2** are the determination of the data of the source system and the development of a Chrysaliser, which transforms the data and transfers it to the target system.
- **Phase** 3: All components except the data are incrementally migrated from the source to the target system.
- In the **4th phase** the data is moved into the target system. Additionally, users are trained in using the new system.
- **Phase 5** deals with the cut-over of the new system (21,25).

1.11 Success factors of data migration

Data migration projects differ tremendously depending on the complexity of the migration. Migrations can be straightforward or complex. 2000 companies spent at least about five billion dollars per year on data migrations and four of five projects need more time or budget than planned. Analyses of successful data migration projects point out the most important factors to achieve a satisfying migration, see figure 1.8 (26,27).

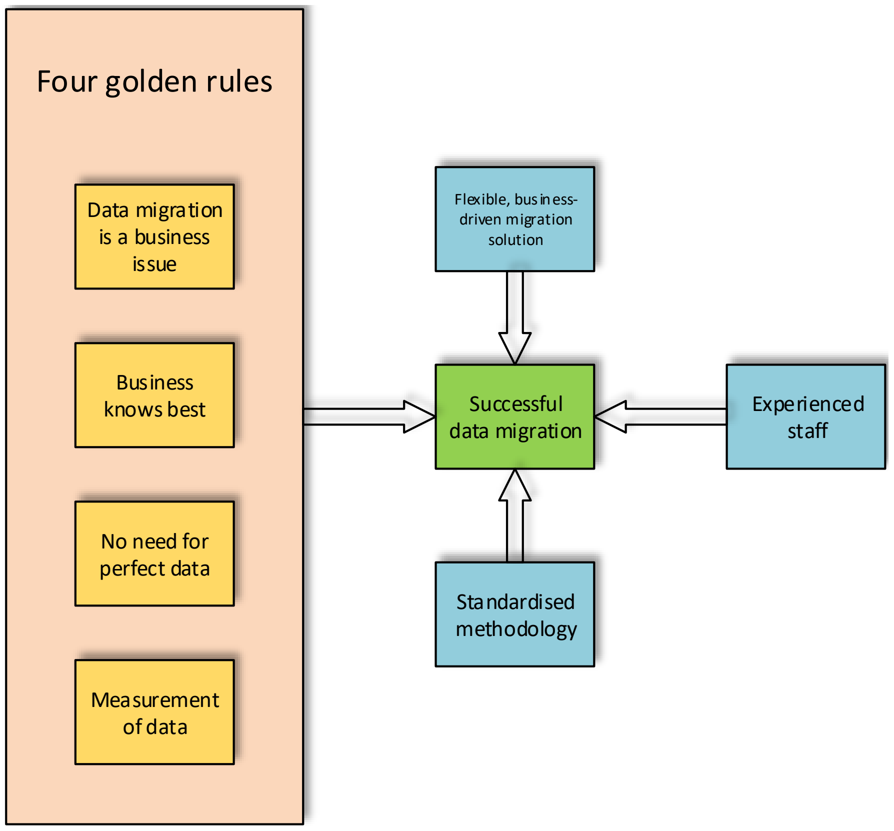


Figure 1.8: Success factors of data migration: Successful data migration is tremendously influenced by the four golden rules: 1) Data migration as a business issue 2) The business knows best 3) No-one needs perfect data 4) If you cannot count it, it does not count. Additionally the experience of the staff, the methodology and the driver influence the success of the migration project (27).

- A data migration is a business issue, because business knows best why data has to be migrated, which data is used in future and when the migration should be completed. If business deals with these issues the technical department can focus on the issue of how data is migrated.
- Business knows best means that the business department should drive the migration. Business stakeholders have to define the requirements and have to take responsibility for the migration project, because they have to know which solution has to be delivered

at which time. The IT department has to handle the problems, which may occur during the migration.

- There is no need for perfect data, instead there is a need for the data required. The goal to store perfect data can cause negative effects in regards to project time and costs. At the project start data owners and users have to define the required quality of data. By knowing the required quality, the IT department can take the appropriate measures to achieve the target data quality.
- A great challenge is the measurement of data. The measurement is important because without it, the quality of data cannot be determined. Measurements have to make sense to the business.
- The scope of the migration project has to be defined clearly.
- Experienced staff and standardized methodologies influence the success of a migration (14,26,27).

1.12 Goal description

The goal of this thesis is the migration of four DBs to one DB, which has to be designed and implemented in Oracle 11g. A MS Access 2010 DB two Oracle 10g DBs and one MySQL DB are the sources. Additionally, XML files have to be migrated to the target system containing data from 2015. All the sources contain data of tissue samples, which are stored in the Biobank Graz, a department of the medical university of Graz.

The tissues in the archive are used for research and the education of students. Scientists have the possibility to request samples and their associated data. Parts of the staff of the Biobank look into its DBs if adequate samples are available. The tissue together with its medical records is very important for a better understanding of disease itself and its cause. Furthermore, new methods for diagnosis, treatment and the prevention of diseases are developed (28).

The DBs contain inter alia information about the gathered tissues e.g. identifiers, description, examination type, date of the examination, type of the extracted organs, etc. It is known that the four source DBs lack on DB design, because the relational aspect does not exist; the tables in the DBs are not related to each other. The two Oracle source DBs and the MySQL DB have not gone productive. Therefore, the stored data has to be analyzed to figure out which kind of data is stored, which information can be extracted out of it and if it is needed in the target system.

1.13Description of the practical task

1.13.1 Main tasks

- Analysis of the four source DBs
- Reconciliation of the data stored in the DBs
- Design and implementation of the target relational DB in Oracle 11g
- Migration of the aligned data of the four source DBs to the target DB
- Migration of the XML files to the target DB

1.13.2 Sub tasks

The main task is the migration of the mentioned DBs in section 1.13.1. The subtasks of this thesis are:

- The migrated data of the source DBs has to be compared and matched.
- Redundancies have to be reduced.
- Analyses have to be made to get an understanding of the use of the productive Access DB.
- The data will be evaluated to be able to decide if it is possible to dismiss not useable data.
- The source data has to be extracted from the Access, Oracle, MySQL DBs and the XML files.
- A DB schema has to be designed. The aim is a typical relational DB schema, where tables refer to each other.
- The loose source constraints have to be reconsidered and adapted. New constraints will be introduced for better DB organization.
- The extracted data from the source has to be transformed in a convenient way. It has to fit into the new designed DB schema.
- A data mapping algorithm has to be designed to copy the selected data from the sources into the target DB.

- Prospectively user requirements analysis will be made to identify in which features and queries the user interface has to cover.

The design of a GUI is not part of this thesis. The design of the GUI will be realized in *Liferay*[1] after the successful completion of the thesis.

1.13.3 Reasons for migration

The reasons for migration between two different RDBMS are various e.g. the upgrade to a new version of the same RDBMS provider, the existing DB is insufficient regarding the performance, economical problems so that the management decides to migrate to a free RDBMS. Another reason could be the unification of different RDBMS to one provider increasing the efficiency and consistency of data. (19)

In the case of the Biobank the Oracle DBMS for the target system has been chosen, because the department wants to unify their DBMS for the mentioned reasons. Currently different DBMS are used e.g. My Structured Query Language (MySQL), Access or Oracle.

The DB has the purpose to be only used in-house. Advantages of such an in-house system are the access can be controlled and adaptions may be made within the organization.

1.14Approach for achieving the migration

At the moment, the source DB are static. A composition of Big Bang and Iterative migration will be used. An Access DB can easily be duplicated by copying the Access file containing the whole DB. With an independent DB file, it is possible to operate without any risk. The data can be moved incrementally to the target Oracle DB without using gateways for synchronization. After the successful migration users may operate on the new system from one day to the next without a complete system shut down. The whole process is not as complex as the data migration process model for business projects, see section 1.5.3, but nevertheless some phases are covered. The following steps are taken to reach the goal of this thesis:

1. Identification of the stored data in the sources
2. Investigation of the sources:

[1] https://www.liferay.com

   a. The meaning of the data and
   b. which data has to be effectively migrated is identified.
   c. Identification of related data within each DB
   d. Identification of commonalities of the DBs
3. Design and implementation of the target Oracle 11g DB.
4. Implementation of the migration process
   a. Execution of data cleansing to identify and cleans redundant, inconsistent or useless data on each of the DBs
   b. Design and implementation of the migration algorithm
      i. Extraction of the data in the sources
      ii. Specification of the data mapping from the extracted data into the target data structure
      iii. Movement of the data
      iv. Test and validation

2 Methods

2.1 Technologies

2.1.1 Java

The term Java comprises two separate things: The Java Virtual Machine (JVM) and the Java programming language. The Java programming language is platform independent. After finishing writing the Java source code it is compiled and transformed into Java byte Code. This byte code consists of a special set of instructions, which can be executed on every platform that is running a Java Virtual Machine (JVM). The JVM interprets the byte code at runtime and executes the program, see figure 2.1. In comparison to programming languages, where the source code is compiled and transformed directly into an executable this kind of execution is slower. Various optimizations of the Java Runtime Environment (JRE) over the years lowered the gap in speed compared to programming languages producing platform-specific executable files. Today, the difference in speed is almost negligible. Java is one of the most popular programming languages. According to PopularitY of Programming Language[1] (PYPL), Java is the most popular programming language. The primary reasons for the popularity of the language are:

- the mentioned platform independency
- provision of applets and servlets on the web
- the object-oriented (OO) paradigm

Additionally, Java is easier to use than other programming languages such as C or C++: Pointers are not implemented, Garbage collection and features for multi-threaded programs are examples for the easier programming in Java (29).

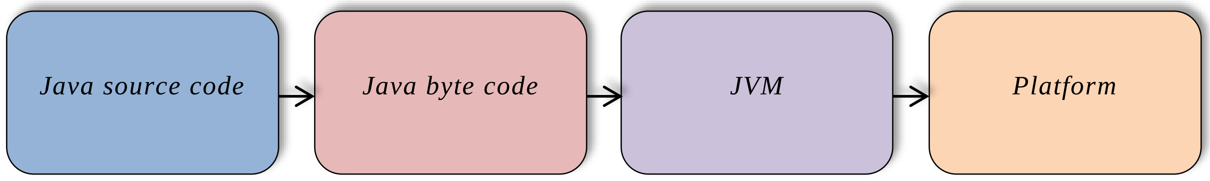


Figure 2.1: Compilation and execution for Java: The Java source code is produced by the programmer. After the source code is finished it gets compiled. The result is Java byte code. The JVM runs and interprets the byte code on the platform (29).

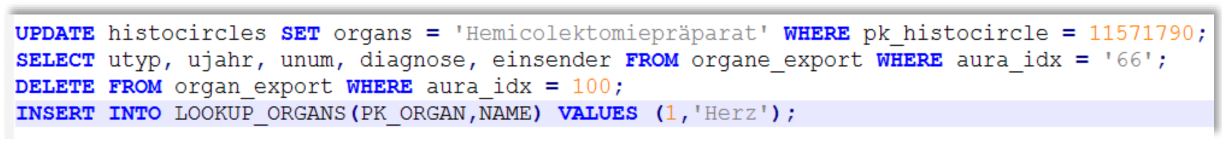
[1] http://pypl.github.io/PYPL.html

2.1.2 Relational Database Management System (RDBMS)

A management system, which is a piece of software, allows the user to insert, delete, modify and retrieve data from a DB. The whole DB represents the stored data. The structure of the DB is organized in tables. Tables consist of columns and rows, where each row represents a data record. Each record can consist of various pieces of information. Each piece of information corresponds to a column of the table. The relational part of the RDBMS allows relating one table to another and joining information from different tables (30).

2.1.3 Structured Query Language (SQL)

SQL is the language to communicate with RDBMSs. It is used to access DBs and to manipulate them. The user writes SQL commands and the RDBMS executes them. SQL is a standardized language of the American National Standards Institute (ANSI). Even though it is standardized, there exist different types. Examples are MySQL, Oracle, Microsoft Access, Microsoft SQL Server, etc. The different RDBMS offer their own extension of the language, which is not transferable to another RDBMS. The different versions of SQL support at least the *INSERT*, *SELECT*, *UPDATE*, *DELETE* and *WHERE* command in a similar way. Listing 2.1 illustrates four of the main SQL commands (31).



```
UPDATE histocircles SET organs = 'Hemicolektomiepräparat' WHERE pk_histocircle = 11571790;
SELECT utyp, ujahr, unum, diagnose, einsender FROM organe_export WHERE aura_idx = '66';
DELETE FROM organ_export WHERE aura_idx = 100;
INSERT INTO LOOKUP_ORGANS(PK_ORGAN,NAME) VALUES (1,'Herz');
```

Listing 2.1 SQL example queries: Each row represents a separate query against the DB. The first query updates the field value of the field "organs" of the table "histocircles". Only records which contain a value of "11571790" in the field "pk_histocircle"are updated. The second SQL statement requests the fields "utyp", "ujahr", "unum", "diagnose" and einsender of the "organe_export" table. Additionally, the user requests only the records of the table, where the field value of the field "aura_idx" contains "66". After executing this request a table containing the query result with the requested fields is returned from the DBMS. The third query deletes every record in the table "organ_export", where the field value of the field "aura_idx" contains "100". The last query inserts a new record into the "lookup_organs" table. The new record contains a value of "1" in the column "PK_ORGAN" and the string value of "Herz" in the column "NAME" (31).

2.1.4 My Structured Query Language (MySQL)

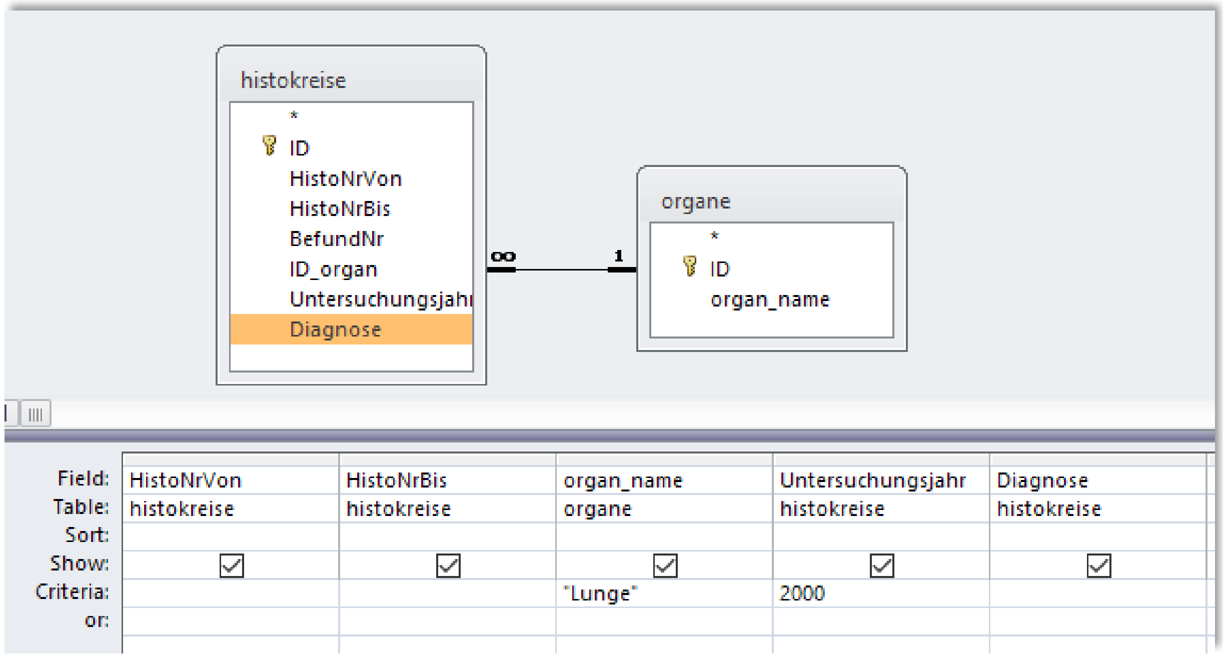
MySQL is an open source RDBMS, which supports a variety of operating systems (OS) such as Windows, Linux and Mac. Four different versions of MySQL are available on the market: The Community Edition, Standard Edition, Enterprise Edition, and Cluster Carrier Grade Edition. Version 5.7 of the Community Edition is used in this thesis and is freely available. MySQL uses SQL, which is the standard query language of modern DB systems, see section 2.1.3. This kind of RDBMS is multi-threaded, which means that multiple client applications, programming interfaces, libraries and administration tools can connect and access the DBs on the server at the same time. The management system offers the possibility to select different engines for storing data, which gives the programmer the opportunity to select the appropriate engine for the desired properties. MySQL offers access control, so that only authorized people can access the data. Various types of programming interfaces, such as Java, PHP, Python, etc. are available. MySQL is the second most popular RDBMS, see section 0 (30,32).

2.1.5 Oracle Database

Oracle is the most popular RDBMS, see section 0. Oracle DB is available in five different versions: Enterprise Edition, Standard Edition, Standard Edition One, Personal Edition and Express Edition. The Express Edition is available for free, but has limited functionality, such as limited memory and no support of Automatic Storage management. Oracle DB supports all of the common OSs, such as Windows, Linux and Mac. Information in an Oracle DB can be accessed by the use of various application programming interfaces (API), such as Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC). Oracle offers the possibility to write programs in the DB by the use of Procedural Language/Structured Query Language (PL/SQL) and Java. Stored procedures, triggers, modules, etc. can be realized within the DB by the use of PL/SQL and Java. Access control is available to ensure that only authorized staff can access the data stored in the DB. In this thesis the Enterprise Editions of the versions 10g and 11g are used on a Windows machine (33,34)

2.1.6 Microsoft Access (MSA)

MSA is an RDBMS including software developing tools for GUI creation. It is part of the Microsoft Office Suite. It is possible to buy MSA separately. Access desktop DBs are files with the extension *accdb*. This file format was introduced with Access 2007. Prior to MSA 2007 the extension was *mdb*. Like Oracle DB and MySQL, MSA uses SQL for DB manipulation. MSA offers the possibility to define queries via a GUI. The advantage of creating queries by using the drag and drop functionality provided by the GUI is that the user does not need to know SQL for less complex requests. The SQL statement is created behind the scenes and can be viewed at the user’s request. Figure 2.2 illustrates a simple graphical query. As a result of the UI, whole DBs can be created without the knowledge of SQL. MSA allows multiple user access. Security issues are addressed by providing data encryption. The software offers the possibility to create forms and reports for data input and output by offering a GUI. Automated actions, which can be used inter alia in forms and reports, can be realized by the use of macros. More complex procedures can be realized by the use of Visual Basic. Access is available for MS Windows as 32 and 64-bit version. In this thesis one of the source DB was provided in MSA 2007 (35).

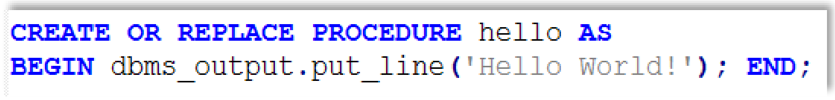


Field:	HistoNrVon	HistoNrBis	organ_name	Untersuchungsjahr	Diagnose
Table:	histokreise	histokreise	organe	histokreise	histokreise
Sort:					
Show:	✓	✓	✓	✓	✓
Criteria:			"Lunge"	2000	
or:					

Figure 2.2: Graphical query generation in MSA: This figure illustrates how a query can be written without the knowledge of SQL. The user selects the tables, where he wants to get the information. Afterwards the desired fields have to be selected by dragging them into the query area. The query can be refined by entering keywords in the criteria field. This graphical MSA query is equal to the SQL command "SELECT h.HistoNrVon, h.HistoNrBis,o.organ_name, h.Untersuchungsjahr, h.Diagnose FROM histokreise h INNER JOIN organe o ON o.ID = h.ID_organ WHERE o.organ_name = 'Lunge' AND h.Untersuchungsjahr = 2000;". Behind the scenes a query like that is constructed in SQL by MSA (35).

2.1.7 Procedural Language/Structured Query Language (PL/SQL)

PL/SQL is a programming language used for Oracle DBs. The language is standardized and portable through Oracle DBs. It is possible to write a procedure on one system and move the same procedure to another compatible DB without any changes of the source code. PL/SQL is optimized for the Oracle DB and can improve the performance of the DB. PL/SQL allows the programmer to execute SQL statements without using APIs such as ODBC, JDBC, etc. PL/SQL is a procedural and OO language. Listing 2.2 illustrates a snippet of example code in PL/SQL (36–38).

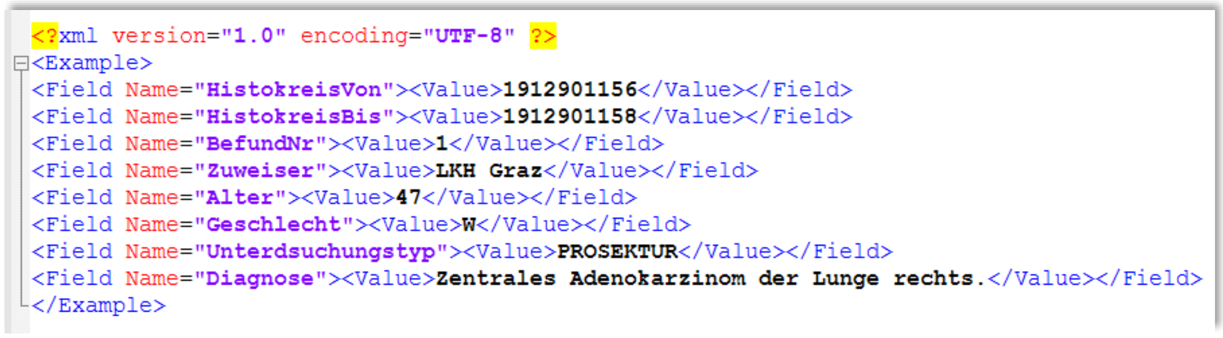


```
CREATE OR REPLACE PROCEDURE hello AS
BEGIN dbms_output.put_line('Hello World!'); END;
```

Listing 2.2: Simple example PL/SQL block: This block represents a procedure, which does not return a value after execution. Instead of returning a value "Hello World!" is shown in the console (36–38).

2.1.8 Extensible Markup Language (XML)

As Hyper Text Markup Language (HTML), which is used to represent web pages, XML is a markup language. As opposed to HTML, XML is not used for viewing data. It is mainly used for structuring and transferring data. The great difference between HTML and XML is that the latter has no fixed number of valid tags, which makes it a meta-markup language. Inter alia, the main objectives of XML are simple usage on the Internet, broad support of applications, compatibility to Standard Generalized Markup Language (SGML), easy processing for machines, easy understanding and readability for users and easy XML document creation. Listing 2.3 illustrates an example XML file (39,40).

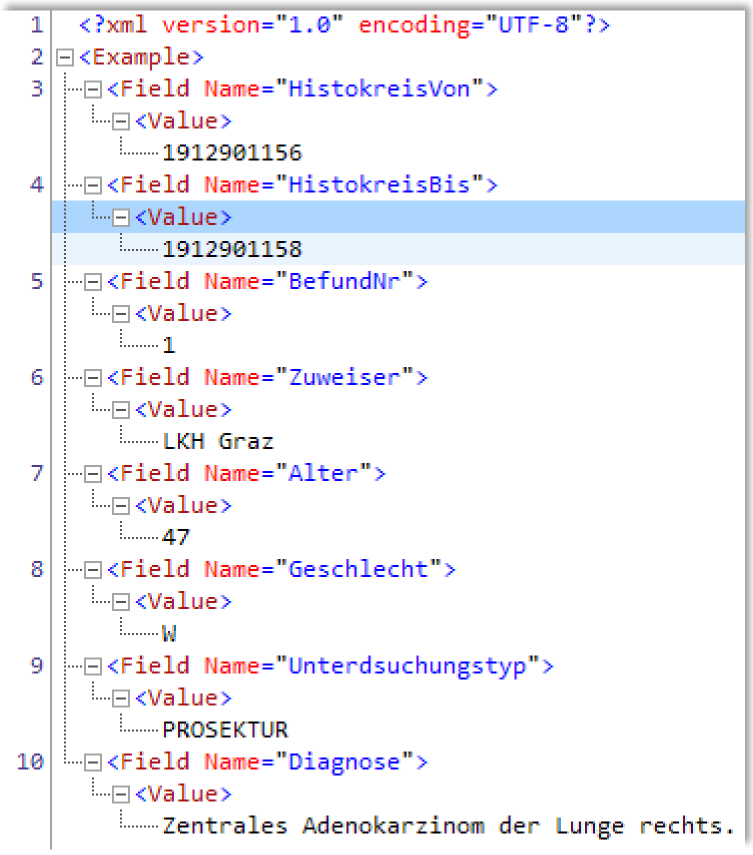


```
<?xml version="1.0" encoding="UTF-8" ?>
<Example>
<Field Name="HistokreisVon"><Value>1912901156</Value></Field>
<Field Name="HistokreisBis"><Value>1912901158</Value></Field>
<Field Name="BefundNr"><Value>1</Value></Field>
<Field Name="Zuweiser"><Value>LKH Graz</Value></Field>
<Field Name="Alter"><Value>47</Value></Field>
<Field Name="Geschlecht"><Value>W</Value></Field>
<Field Name="Unterdsuchungstyp"><Value>PROSEKTUR</Value></Field>
<Field Name="Diagnose"><Value>Zentrales Adenokarzinom der Lunge rechts.</Value></Field>
</Example>
```

Listing 2.3: XML example file structure: The first line defines the xml version and the encoding of the XML file. In the following lines "Field" tags are defined with the attributes "HistokreisVon", "HistokreisBis", "BefundNr", etc. Within this container the values for these fields are defined. In the sixth line for example somebody's age is defined by 47- probably years (39,40).

2.1.9 Document Object Model (DOM)

The DOM is a cross platform API, which is available for any programming language such as Java, PHP: Hypertext Preprocessor (PHP), C++, Scala, etc. The DOM is the most well-known DM for processing and storing XML and HTML documents. The DOM represents the logical structure and how information is accessed and manipulated in these kinds of files. Within this thesis the DOM of XML files is accessed to migrate them into the Oracle DB. Figure 2.3 illustrates the DOM of the XML document in listing 2.3 (41,42).



```
<?xml version="1.0" encoding="UTF-8"?>
<Example>
    <Field Name="HistokreisVon">
        <Value>
            1912901156
    <Field Name="HistokreisBis">
        <Value>
            1912901158
    <Field Name="BefundNr">
        <Value>
            1
    <Field Name="Zuweiser">
        <Value>
            LKH Graz
    <Field Name="Alter">
        <Value>
            47
    <Field Name="Geschlecht">
        <Value>
            W
    <Field Name="Unterdsuchungstyp">
        <Value>
            PROSEKTUR
    <Field Name="Diagnose">
        <Value>
            Zentrales Adenokarzinom der Lunge rechts.
```

Figure 2.3: DOM XML tree: This figure shows the DOM tree of the XML document in Listing 2.3. "Example" is the root node. The root node is followed by its children the "Field" elements containing a child named "Value" (41,42).

2.2 Tools

2.2.1 Git

Git is a Distributed Version Control System (DVCS), which is used to support software development in this thesis. DVCS systems fully mirror the complete repository of a server. This has the advantage that the repository can be completely rebuilt by the clients if the server collapses, because each of the clients is a complete backup. Most of the other version control systems, such as *Subversion* store the differences of the base file after each commit. In contrast Git always stores whole files for each version and lots of functions are locally executable. Lots of clients are available for different OSs and Integrated Development Environments (IDE), such as Eclipse, NetBeans, IntelliJ IDEA, etc. Git is available for free and for the following OSs: Windows, Mac OS X, Linux and Solaris (43).

2.2.2 TortoiseGit

TortoiseGit is a graphical Windows client for the Git DVCS, see section 2.2.1. It is freely available. The tool is integrated into the Windows shell e.g. the Explorer. There it is possible to view the status of the Git project. In the context menu of the Explorer it is possible to execute various Git commands, such as commit, delete, showing the log tree, etc. (44).

2.2.3 Dia

Dia is a free diagram editor for Windows, Mac OS X and Linux. It offers various shape packages, which support the drawing of different diagrams and charts, such as network layouts, flowcharts, entity relationship models (ERM), UML diagrams, etc. Beside the native format Dia supports additional ones such as *png*, *svg*, *tex*, *cgm*, etc. Dia provides an easy to understand UI and is used for the creation of the ERM of this thesis (45).

2.2.4 Mindfusion XML viewer

This XML software is available for free and exclusive to the Windows platform. Beside viewing XML files, the program allows the user to create and modify XML documents. Instead of viewing the document in the standard editor view, the XML file is presented in a hierarchical tree form equivalent to the DOM, see section 2.1.9. If an XML file is not valid, e.g. if a tag is not closed, the software refuses to open the document (46,47).

2.2.5 phpMyAdmin

phpMyAdmin is a free browser-based DB administration tool, which helps managing MySQL DBs. It provides an intuitive GUI. With the UI of phpMyAdmin it is easy to create, modify, query and import DBs, tables, data records, fields and more. All types of queries can be executed by using the integrated SQL editor. Additionally, simple but useful queries such viewing, creating or sorting tables are already integrated in the UI and can easily be executed by a single click of the mouse button (48).

2.2.6 Notepad++

Notepad++ is a free notepad and source code editor available for the Windows OS. The software is written in C++. The software offers high execution speed by using less resources of the central processing unit (CPU). Additionally, the consumption of memory is low. Notepad++ supports syntax highlighting and syntax folding for lots of programming languages and other languages including SQL, Java, C, XML, etc. The community provides several plugins, extending support of languages und functionality of the software. Multi- and column mode editing allow a quick modification of the file by offering the possibility to edit multiple rows at the same time. Useful tools, such as regular expression (REGEX) search and replace functions, help to edit the source code more efficient. During this thesis, Notepad++ is used for Syntax highlighting of the MSA SQL source code, because MSA does not provide this feature in version 2010. Additionally it is used for quick modification of text files by using REGEX (49).

2.2.7 SQL Workbench/J

SQL Workbench/J is a cross platform SQL client, which can connect to various DBMS, such as Oracle DB, MySQL and PostgreSQL. It can be executed on Windows, Linux, Mac and other OS where the JRE is running. The connection to the DBs can be established via the JDBC driver, which has to be configured in SQL Workbench/J. Important features are the execution of SQL queries. The requested data can be edited in the result of the query. The client offers comprehensive import and export functions e.g. XML, CSV or SQL files can be easily generated to export data. The Data Pumper of SQL Workbench/J allows the user to copy data from one table of a source DB to another of a target DB. The SQL client supports important features such as auto completion for tables and columns, syntax highlighting, auto formatting for SQL statements. SQL Workbench/J is free software and should not be mixed with *MySQL Workbench* providing inter alia similar features (50).

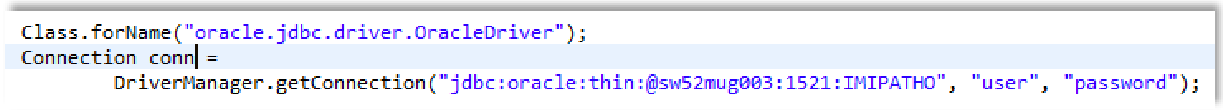
2.2.8 Eclipse IDE

Eclipse IDE is a cross platform IDE for software development. It is free and available for Windows, Linux and Mac. One of the core concepts of Eclipse is the modular architecture. This feature allows to add or delete components, called “Plugins” or “Bundles”. The available plugins allow the user to adapt the platform individually to the user’s needs. A popular example are the C/C++ Developer Tools, which are competing head to head with Visual Studio for the most popular IDE for C. Eclipse supports lots of different programming languages by installing the corresponding plugins. Eclipse is a tool platform as well, which means that there are hundreds of tools available such as data tools for DB administration, web tools for web developers, but also tools which are not part of software development like business intelligence and reporting tools providing graphical and text analysis. Eclipse is an open-source platform, which means that the source code is visible for everybody (51,52).

2.3 Libraries and plugins

2.3.1 Java Database Connectivity (JDBC)

The JDBC package may be the most prominent API for gaining DB access from Java. It supports SQL functionality for lots of RDBMS. The API is used in Java programs communicating with RDBMS. To gain access to a DB in Java via the JDBC API, see listing 2.4 the driver is registered for usage. After initializing the *DriverManager* the class loads the driver. Afterwards a connection to the DB can be established, which offers the possibility to send requests to the RDBMS. A connection can look like the following example: *jdbc:oracle:thin//server:3306/db* (2).



```
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection conn =
        DriverManager.getConnection("jdbc:oracle:thin:@sw52mug003:1521:IMIPATHO", "user", "password");
```

Listing 2.4: Example DB connection in Java via JDBC: The first line registers the driver. The follwing lines define a DB connection with the registered Oracle JDBC driver. The protocol "oracle" and subprotol "thin" is defined. In this example the subname consists of the sever "sw52mug003", the port "1521" and the Oracle instance with the name "IMIPATHO". Additionally the DB connection needs the username and the password to communicate with the RDBMS (2).

2.3.2 Open Database Connectivity (ODBC)

The ODBC standard was produced by Microsoft and is used for accessing heterogeneous SQL DBMS. The API allows to communicate to different DBs by the use of the same source code. Client – server software can be written, accessing various types of DBMS. The API defines inter alia library functions for connection to the DB, execution of queries and retrieval of results. Additionally, standard set of errors, standard representation of data types and a standard connection to the DBs are defined. ODBC is very popular and is de facto industry standard. ODBC DB drivers are available for the most popular DBMSs such as Oracle DB and MySQL (2).

3 Results

In this section the implementation of a generic data migration process is described in detail, see section 3.5. Additionally, the contents of the sources are presented and explained, see section 3.1 – section 3.3. Furthermore, the newly implemented target DB schema is explained in detail, see section 3.4.

3.1 Content explanation of the sources

The DBs and the XML files contain data of tissue samples. The samples of tissue are available in blocks or slides, see figure 3.1. A slide is produced by cutting a thin slice of the block. Information about the blocks and slides of the samples is stored in a system called *Blocktracking*. The system holds inter alia information about the place and position of storage.

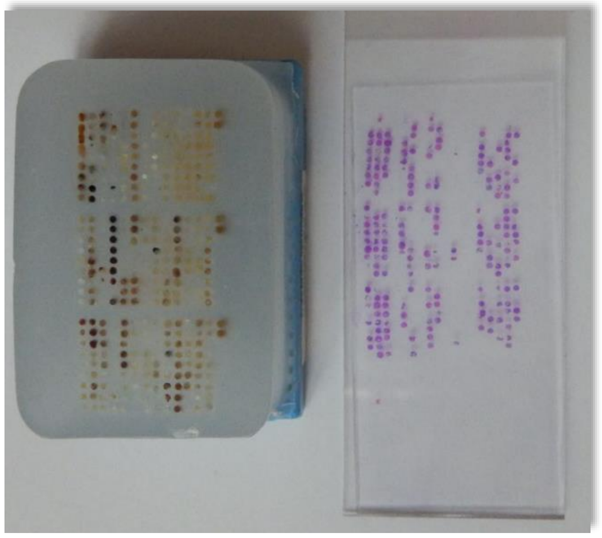


Figure 3.1: Photo of an FFPE block and slide: The photo illustrates a FFPE block on the left and a slide of the block on the right.

In general, the blocks and slides of tissue get to the Biobank by the following steps

1. Surgery is performed in various clinical institutions of Styria.
2. Tissue removed during surgery is transported to the institute of pathology.
3. The pathologist makes the exact histopathological diagnosis.
4. The tissue is stored in the Paraffin archive of the Biobank Graz. All the samples get barcoded.

There are two different types of tissue samples stored in the archive of the Biobank Graz:

- Formalin-Fixed Paraffin-Embedded (FFPE) tissue
- Fresh frozen tissue

FFPE tissue samples are unselected pathological samples that represent all detected diseases at their natural frequency of occurrence. Cross-sectional biobanks store unselected pathological samples. The Biobank Graz stores the FFPE tissue samples in cooperation with the institute of Pathology (53).

The tissue is immersed in formalin for a specific time to achieve a fixation. Fixation is the technical term for preservation of the components of tissues and cells. The fixed tissue is dehydrated by the use of dimethyl-benzene and ethanol series and embedded in a layer of paraffin wax. The result of this process is an FFPE block which can be stored at room temperature for decades and longer (54–57).

The fresh frozen tissues are snap frozen by the use of liquid nitrogen (LN2) and stored at a temperature of -196 °C. Disease focused clinical biobanks store samples of tissue in that way to be able to observe e.g. diseases of interest (53,58).

The blocks and slides of tissue are attached to patient cases. A case is associated with the surgery of the patient. The standard procedure for classification of tissues at the Biobank of Graz is that the cut out tissues are grouped and a number is assigned to each group called *Histologic number* (HNO). The first group receives the lowest number, the next receives the next higher number etc. The groups assigned to one case are called *Histologic circle* (HCI). The histopathological diagnoses are also assigned to numbers. The lowest number is assigned to the first, containing the main diagnose. The following diagnoses receive the next higher numbers. In the end a HCI can contain various HNOs and various numbers of histopathological diagnosis. The tables in the sources contain the

- Information about the cases (HCIs)
- Information about the finding (HNO)

in one dataset. Each HNO belongs to an HCI. Table 3.1 illustrates which type contains which content.

Information about the case (HCI)	**Information about the finding (HNO)**
First histologic number	Examination type
Last histologic number	Date of the examination
Examination year	Pathologic diagnose
Department	Histologic result
Organ	Macroscopic result
Patients forename	Underlying disease
Patients surname	Additional disease
Patients date of birth	Coroner’s inquest
Patients age	International Classification of Diseases (ICD) – O Code
Patients sex	ICD – 10 Code
Sender	Additional Parameters such as tumor staging parameters and Receptor tyrosine-protein kinase erbB-2 (HER2) parameters

Table 3.1: Overview of the types of information about the sources: The information about the sources can be divided up into the information about the case (HCI) and the finding (HNO). A finding always belongs to a case and contains specific information about the diagnosis.

3.2 Identification of the stored data in the source databases

At the beginning of this thesis the data contained in the source DBs is pretty unknown. Thus, the first task is the identification of the data stored in the different DBs to understand which data is relevant for future use. Thus, an investigation is started to identify the data. Documentations and interviews help to identify the unknown data. The gathered information about the different sources is presented, see section 0 – section 3.2.5. The fields marked with the tick have to be migrated and the fields marked with the red cross not. This information is relevant for the profile and audit phase of the migration, see section 3.5.2. Additionally, aliases are introduced for an easier understanding of the further process, see section 3.2.6.

All the mentioned sources contain various information about samples of tissues. Inter alia information about the extracted organs, patient sex and pathologic diagnosis are stored within the sources. Each sample belongs to an HNO. The datasets contain information about the cases respectively HCIs, which consist of one or more continuous HNOs. The HCI number is always identified by the lowest contained HNO.

3.2.1 Microsoft Access DB

The MSA DB is stored as *tdb.mdb* and is used by the project management of the Biobank. Documentations and interviews with the staff of the project management and an employee of the Biobanking and BioMolecular resources Research Infrastructure (BBMRI) help to extract information out of the data.

Group No	**Tables**
✓ *1*	Organe 01/01/1984 – 30/06/2006
✓ *2*	ALLE METASTASE, CA+Colotis, Colitis Ulcerosa, COLITIS+Patienten, KOLON CA DIE META:HABEN, KOLON CA_NEU, KOLON META OHNE DUPLIKATEN, Mamma metastase nach 2006 + Primär CA, METASTASEN VON CRC PATIENTEN, Patienten HarnblasePrim+Metastase
✓ *3*	2001-2004, 2003-2005, Arthritis, astrozytom, Cholangiozellu, CRC 2001-2004, crc in leber, CRC Lebermeta, Dermatomyositis, Endometrium CA, Fibröse Dysplasie, FISH *****, FISH 2005-2006, GIST, Glioblast, hepatocelll, HER 2 NEU (3+), her2, hirn, histo-zytom, Hoden, keinklein, leber biopsie, LIPOSARKOM 2003, LUNGEN CA ALLE SEB, Malignes Melanom, mamma 200, Mamma ab 2000 alles DG, Mamma CA+Meta HER2/Neu, Mamma meta, Mamma meta 2001-2002, Mamma meta 2001-2002 tubul, mamma normal gewebe, MammaCa, medulloblastom, Metastase Harnblase CA, mitellhirn, ohneadenocervix, onko erweiterung, onko2, onkozyt, onkozytom, Ovarial CA, Pankreas, Plasmozytom, Pleomorphe liposarkom, Prostata mach, ProstataCA, Rhabdomyosarkom, Rückenmark, score 0 und 1, score 2, score3, sentinel11, Wilms Tumor
✓ *4*	Nase, Nasenpolype
✓ *5*	NSCLC, mamma ca, , HCC aus 2006
✓ *6*	FISH Ausgewählt
✓ *7*	BefundeAnn2006_2009
✗ *8*	Asslaber histonum, HarnBl Ca, HistoPenn, Kolon Ca Cryo-Patienten, Prostata

Table 3.2: Grouping of the 79 tables contained in the MSA DB. The groups from one to seven contain important data. The data stored in the eighth group is not useable because of lack of information and is not listed in table 3.3.

The DB contains 79 tables. The tables are not related to each other. The main table of the DB is *Organe 01/01/1984 – 30/06/2006* and is used by the project management to search for tissue samples within the DB. After the investigation of the tables it is obvious that 74 of them contain important data and five data that is not useable. The tables are grouped by their containing fields and usability, see table 3.2. The tables contain information about the examination date, diagnoses, etc., see table 3.3.

Field name	**Data type**	**Description**	**Groups**
✓ *Aura_Idx*	Short text	The fields *uTyp*, *uDate* and *uNum* are concatenated, followed by the *supplement number* of the case. The last has to be migrated.	1, 3, 5, 6
✓ *uTyp*	Short text	Contains the abbreviation of the examination type, which describes the process of extraction	1, 2, 3, 4, 5, 6
✓ *uDate*	Date / Time	Contains the extraction date of the tissue	1, 2, 3, 4, 5, 6
✓ *uJahr*	Number	Contains the year when the tissue was extracted	1, 2, 3, 4, 5, 6
✓ *uNum*	Number	First number of the admission	1, 2, 3, 4, 5, 6
✓ *Es_Anz*	Number	Number of tissues which where extracted	1, 2, 3, 4, 5, 6
✓ *Nachname*	Short text	Contains the patients surname	1, 2, 3, 4, 5, 6
✓ *Vorname*	Short text	Contains the patients first name	1, 2, 3, 4, 5, 6
✓ *Pat_Geb*	Date / Time	Date of birth of the patient	1, 2, 3, 4, 5, 6
✓ *OR*	Short text	Contains the organs whereof the tissues were extracted	1, 2, 3, 4, 5, 6
✓ *Diagnose*	Long text	Contains the pathologic diagnosis of the pathologist	1, 2, 3, 4, 5, 6
✓ *Einsender*	Short text	Contains information about the clinic respectively the doctor who sent the tissues	1, 2, 3, 6
✓ *Uterus*	Number	Boolean value, generated automatically by a text mining software. A “1” means that the case contains this organ.	1
✓ *Ovar*	Number	See *Uterus*	1
✓ *Magen*	Number	See *Uterus*	1
✓ *Dünndarm*	Number	See *Uterus*	1
✓ *Dickdarm*	Number	See *Uterus*	1
✓ *Appendix*	Number	See *Uterus*	1
✓ *Leber*	Number	See *Uterus*	1
✓ *Mamma*	Number	See *Uterus*	1
✓ *Haut*	Number	See *Uterus*	1
✓ *Prostata*	Number	See *Uterus*	1

✓ *Schilddrüse*	Number	See *Uterus*	1
✓ *Ösophagus*	Number	See *Uterus*	1
✓ *Lunge*	Number	See *Uterus*	1
✓ *Niere*	Number	See *Uterus*	1
✓ *Harnwege*	Number	See *Uterus*	1
✓ *Pancreas*	Number	See *Uterus*	1
✓ *ZNS*	Number	See *Uterus*	1
✓ *Lymphknoten*	Number	See *Uterus*	1
✓ *Knochenmark*	Number	See *Uterus*	1
✓ *Herz*	Number	See *Uterus*	1
✓ *Befundnummer*	Number	First number of the case, containing the year	7
✓ *Nb*	Number	The supplement number of diagnostic finding	7
✓ *Bisnummer*	Number	Last number of the case	7
✓ *Untjahr*	Number	See *uJahr*	7
✓ *Pag_age*	Number	The age of the patient	7
✓ *Unttyp*	Short text	See *uTyp*	7
✓ *Pat_sex*	Short text	The sex of the patient	7
✓ *Material*	Short text	See *OR*	7
✓ *Histo*	Long text	The histologic result	7
✓ *Makro*	Long text	The macroscopic result	7
✓ *Grundleiden*	Short text	The underlying disease of the patient	7
✓ *Nebenleiden*	Short text	Additional diseases of the patient	7
✓ *Todesursache*	Short text	The coroner's inquest of the patient	7
✗ *Materialkuerzel*	Short text	Shortcut of the field *Material*	7
✗ *Mp_beschreibung*	Short text	Molecular pathologic diagnosis	7
✗ *Np_befund*	Short text	Macroscopic description	7

✓ *Grad*	Short text	Parameter of the tumor staging	7
✓ *Tnmp*	Short text	See *Grad*	7
✓ *Tnmt*	Short text	See *Grad*	7
✓ *Tnmm*	Short text	See *Grad*	7
✓ *Tnmn*	Short text	See *Grad*	7
✓ *Tnmr*	Short text	See *Grad*	7
✓ *Tnml*	Short text	See *Grad*	7
✓ *Tnmv*	Short text	See *Grad*	7
✓ *Dg_code1*	Short text	Contains the ICD O Codes of the case	7
✓ *Dg_code2*	Short text	Contains the ICD 10 Codes of the case	7
✗ *Pap*	Short text	Contains the results of the Papanicolaou (PAP) test detecting cancerous processes in the cervix.	7
✗ *Gyn_qualität*	Short text	Gynecological description	7
✓ *Bereich*	Short text	Contains the abbrevation of the department wherefrom the tissues were sent	7
✗ *Disease_id*	Number	Data generated by the text mining software, which extracts the ICD O, ICD 10 and the parameters of the Tumor staging out of the data of the case	7
✗ *Patient_id*	Short text	See *Disease_id*	7
✗ *Finding_id_diagnosen*	Short text	See *Disease_id*	7
✗ *Time_diff*	Short text	See *Disease_id*	7
✗ *Age*	Short text	See *Disease_id*	7
✗ *Diagnosis*	Long text	See *Disease_id*	7
✗ *Diagnosis clean*	Long text	See *Disease_id*	7
✗ *Organ_zuordnung*	Short text	See *Disease_id*	7
✗ *Doctor*	Short text	See *Disease_id*	7
✗ *Ofs*	Short text	See *Disease_id*	7
✗ *Sender*	Short text	See *Disease_id*	7
✗ *Sender_typ*	Short text	See *Disease_id*	7

✗ *Feld 45*	Short text	See *Disease_id*	7
✗ *Morphologyclass*	Short text	See *Disease_id*	7
✗ *Morphologyid*	Short text	See *Disease_id*	7
✗ *Topologyid*	Short text	See *Disease_id*	7
✗ *Dict_id_mor*	Short text	See *Disease_id*	7
✗ *Dict_id_top*	Short text	See *Disease_id*	7
✗ *T*	Short text	Parameter of the Tumor staging, automatically generated by the text mining software	7
✗ *G*	Short text	See *T*	7
✗ *M*	Short text	See *T*	7
✗ *R*	Short text	See *T*	7
✗ *N*	Short text	See *T*	7
✗ *L*	Short text	See *T*	7
✗ *V*	Short text	See *T*	7
✗ *SCG*	Short text	See *Disease_id*	7
✗ *Organ_test*	Short text	See *Disease_id*	7
✗ *Average_examina-tion_date*	Short text	See *Disease_id*	7
✗ *Examination_typ*	Short text	See *Disease_id*	7
✗ *Imi_id*	Short text	See *Disease_id*	7
✗ *Geticd10codes2*	Short text	See *T*	7
✗ *Geticdocodes2*	Short text	See *T*	7
✓ *HER2neu/CEP17*	Short text	HER2 parameter	6
✓ *RATIO*	Short text	See *HER2neu/CEP17*	6
✓ *Her2Neu(DAKO)*	Short text	See *HER2neu/CEP17*	6
✓ *RATIO VOLL*	Short text	See *HER2neu/CEP17*	6

Table 3.3: Description of the fields contained in all the tables of the MSA DB. The tables are assigned to groups, see table 3.2. The groups are assigned to the fields of the DB. The content or parts of the content of the fields marked with the tick are migrated to the target DB.

3.2.2 Oracle 10g database 1

The System Identifier (SID) of this Oracle DB is *Imirep*. The DB contains two unrelated tables containing data fields, which are known from the tissue DB, see section 0.

Identifier	**Tables**
1	Fall_data
2	Befunde20062009

Table 3.4:Assigned IDs of the tables contained in the Imirep DB.

To guarantee a better overview each of the tables is assigned to an identifier, see table 3.4. Table 3.5 gives an overview and a description of the fields contained in the tables of the DB.

Field name	**Data type**	**Description**	**Groups**
✓ *Aura_idx*	Varchar2(20)	The fields *uTyp*, *uDate* and *uNum* are concatenated, followed by the *supplement number* of the case. The last has to be migrated.	1
✓ *Utyp*	Varchar2(2)	Contains the abbreviation of the examination type, which describes the process of extraction	1
✓ *Udate*	Varchar2(20)	Contains the extraction date of the tissue	1
✓ *Uyear*	Number(4)	Contains the year when the tissue was extracted	1
✓ *Unum*	Number(10)	First number of the admission	1
✓ *Esanz*	Number(3)	Number of tissues which where extracted	1
✓ *Nachname*	Varchar2(30)	Contains the patients surname	1
✓ *Vorname*	Varchar2(30)	Contains the patients first name	1
✓ *Pat_Geb*	Varchar2(20)	Date of birth of the patient	1
✓ *Org*	Varchar2(100)	Contains the organs whereof the tissues were extracted	1
✓ *Diagnose*	Varchar2(4000)	Contains the pathologic diagnosis of the pathologist	1
✓ *Einsender*	Varchar2(100)	Contains information about the clinic respectively the doctor who sent the tissues	1
✓ *Uterus*	Number(1)	Boolean value, generated automatically by a text mining software. A "1" means that the HCI contains this organ.	1
✓ *Ovar*	Number(1)	See *Uterus*	1
✓ *Magen*	Number(1)	See *Uterus*	1

✓	*Duenndarm*	Number(1)	See *Uterus*	1
✓	*Dickdarm*	Number(1)	See *Uterus*	1
✓	*Appendix*	Number(1)	See *Uterus*	1
✓	*Leber*	Number(1)	See *Uterus*	1
✓	*Mamma*	Number(1)	See *Uterus*	1
✓	*Haut*	Number(1)	See *Uterus*	1
✓	*Prostata*	Number(1)	See *Uterus*	1
✓	*Schilddruese*	Number(1)	See *Uterus*	1
✓	*Oesophagus*	Number(1)	See *Uterus*	1
✓	*Lunge*	Number(1)	See *Uterus*	1
✓	*Niere*	Number(1)	See *Uterus*	1
✓	*Harnwege*	Number(1)	See *Uterus*	1
✓	*Pancreas*	Number(1)	See *Uterus*	1
✓	*ZNS*	Number(1)	See *Uterus*	1
✓	*Lymphknoten*	Number(1)	See *Uterus*	1
✓	*Knochenmark*	Number(1)	See *Uterus*	1
✓	*Herz*	Number(1)	See *Uterus*	1
✓	*Befundnummer*	Varchar2(20)	First number of the case, containing the year	2
✓	*Nb*	Varchar2(20)	The supplement number of the diagnostic finding	2
✓	*Bisnummer*	Varchar2(20)	Last number of the case	2
✓	*Untjahr*	Varchar2(20)	See *uJahr*	2
✓	*Pag_age*	Varchar2(20)	The age of the patient	2
✓	*Unttyp*	Varchar2(20)	See *uTyp*	2
✓	*Pat_sex*	Varchar2(20)	The sex of the patient	2
✓	*Material*	Varchar2(200)	Contains the organs whereof the tissues were extracted	2

✓ *Histo*	CLOB	The histologic result	2
✓ *Makro*	CLOB	The macroscopic result	2
✓ *Grundleiden*	CLOB	The underlying disease of the patient	2
✓ *Nebenleiden*	CLOB	Additional diseases of the patient	2
✓ *Todesursache*	CLOB	The coroner’s inquest of the patient	2
✗ *Materialkuerzel*	Varchar2(20)	Abbreviation of the field *Material*	2
✗ *Mp_beschreibung*	CLOB	Molecular pathologic diagnosis	2
✗ *Np_befund*	CLOB	Macroscopic description	2
✓ *Grad*	Varchar2(20)	Parameter of the tumor staging	2
✓ *Tnmp*	Varchar2(20)	See *Grad*	2
✓ *Tnmt*	Varchar2(20)	See *Grad*	2
✓ *Tnmm*	Varchar2(20)	See *Grad*	2
✓ *Tnmn*	Varchar2(20)	See *Grad*	2
✓ *Tnmr*	Varchar2(20)	See *Grad*	2
✓ *Tnml*	Varchar2(20)	See *Grad*	2
✓ *Tnmv*	Varchar2(20)	See *Grad*	2
✓ *Dg_code1*	Varchar2(20)	Contains the ICD O Codes of the case	2
✓ *Dg_code2*	Varchar2(20)	Contains the ICD 10 Codes of the case	2
✗ *Pap*	Varchar2(20)	Contains the results of the PAP test detecting cancerous processes in the cervix.	2
✗ *Gyn_qualität*	Varchar2(20)	Gynecological description	2
✓ *Bereich*	Varchar2(20)	Contains the shortcut of the department wherefrom the tissues were sent	2
✗ *Disease_id*	Varchar2(20)	Data generated by the text mining software, which extracts the ICD O, ICD 10 and the parameters of the Tumor staging out of the data of the case	2
✗ *Finding_id_diagnosen*	Varchar2(20)	See *Disease_id*	2
✗ *Time_diff*	Varchar2(20)	See *Disease_id*	2
✗ *diagnosis clean*	CLOB	See *Disease_id*	2

✗ *Organ_zuordnung*	Short text	See *Disease_id*	2
✗ *Organ*	Varchar2(180)	See *Disease_id*	2
✗ *Doctor*	Varchar2(20)	See *Disease_id*	2
✗ *Ofs*	Varchar2(20)	See *Disease_id*	2
✗ *Morphologyclass*	Varchar2(20)	See *Disease_id*	2
✗ *Morphologyid*	Varchar2(20)	See *Disease_id*	2
✗ *Topologyid*	Varchar2(20)	See *Disease_id*	2
✗ *Topologyclass*	Varchar2(20)	See Disease_id	2
✗ *Dict_id_mor*	Varchar2(20)	See *Disease_id*	2
✗ *Dict_id_top*	Varchar2(20)	See *Disease_id*	2
✗ *T*	Varchar2(20)	Parameter of the Tumor staging, automatically generated by the text mining software	2
✗ *G*	Varchar2(20)	See *T*	2
✗ *M*	Varchar2(20)	See *T*	2
✗ *R*	Varchar2(20)	See *T*	2
✗ *N*	Varchar2(20)	See *T*	2
✗ *L*	Varchar2(20)	See *T*	2
✗ *V*	Varchar2(20)	See *T*	2
✗ *Examination_typ*	Varchar2(20)	See *Disease_id*	2
✗ *Imi_id*	Varchar2(20)	See *Disease_id*	2
✗ *Geticd10codes2*	Varchar2(100)	See *T*	2
✗ *Geticdocodes2*	Varchar2(50)	See *T*	2
✗ *Reviewed_from*	Varchar2(20)	See *Disease_id*	2
✗ *Review*	Varchar2(50)	See *Disease_id*	2
✓ *Einsender*	Varchar2(20)	Contains information about the clinic respectively the doctor who sent the tissues	2

Table 3.5: Description of the fields contained in all the tables of Imirep. The tables are assigned to IDs, see table 3.4. The IDs are assigned to the fields of the DB. The content or parts of the content of the fields marked with the tick are migrated to the target DB.

3.2.3 Oracle 10g database 2

The SID of the second Oracle DB is *Imipatho*. The DB contains three tables and two views. The table called plan_table contains no data. The two views refer to not existing tables and therefore they are not usable. Table 3.6 lists every object of the DB and assigns them to an ID.

Identifier	**Name**	**Type**
1	Imi_aura_data	Table
2	Imi_grz_data	Table
3	Plan_table	Table
4	Patho_bb_aura	View
5	Patho_bb_grz	View

Table 3.6: Overview of the objects contained in the Imipatho DB. The IDs are introduced to get a better overview in Table 3.7.

Table 3.7 gives a description of the fields of the tables and their description.

Field name	**Data type**	**Description**	**Groups**
✓ *Befundnummer*	Varchar2(8)	First number of the case, containing the year	1
✓ *Nb*	Varchar2(2)	The supplement number of diagnostic finding	1
✓ *Bisnummer*	Varchar2(2)	Last number of the case	1
✓ *Untjahr*	Varchar2(4)	See *uJahr*	1, 2
✓ *Pag_age*	Varchar2(3)	The age of the patient	1, 2
✓ *Unttyp*	Varchar2(2)	Contains the abbreviation of the examination type, which describes the process of extraction	1
✓ *Pat_sex*	Varchar2(1)	The sex of the patient	1, 2
✓ *Material*	Varchar2(100)	Contains the organs whereof the tissues were ex-tracted	1
✓ *Einsender*	Varchar2(14)	Contains information about the clinic respectively the doctor who sent the tissues	1, 2
✓ *Diagnose*	CLOB	Contains the pathologic diagnosis of the pathologist	1, 2
✓ *Histo*	CLOB	The histologic result	1, 2
✓ *Makro*	CLOB	The macroscopic result	1, 2
✓ *Grundleiden*	CLOB	The underlying disease of the patient	1, 2

✓ *Nebenleiden*	CLOB	Additional diseases of the patient	1, 2
✓ *Todesursache*	CLOB	The coroner's inquest of the patient	1, 2
✓ *Befundnummer*	Varchar2(10)	First number of the case, containing the year	1
✓ *Nb*	Varchar2(1)	The supplement number of diagnostic finding	1
✓ *Bisnummer*	Varchar2(10)	Last number of the case	1
✓ *Unttyp*	Varchar2(3)	See *uTyp*	1
✓ *Material*	Varchar2(200)	Contains the organs whereof the tissues were extracted	2
✗ *Materialkuerzel*	Varchar2(10)	Abbreviation of the field *Material*	2
✗ *Mp_beschreibung*	CLOB	Molecular pathologic diagnosis	2
✗ *Np_befund*	CLOB	Macroscopic description	2
✓ *Grad*	Varchar2(20)	Parameter of the tumor staging	2
✓ *Tnmp*	Varchar2(20)	See *Grad*	2
✓ *Tnmt*	Varchar2(20)	See *Grad*	2
✓ *Tnmm*	Varchar2(20)	See *Grad*	2
✓ *Tnmn*	Varchar2(20)	See *Grad*	2
✓ *Tnmr*	Varchar2(20)	See *Grad*	2
✓ *Tnml*	Varchar2(20)	See *Grad*	2
✓ *Tnmv*	Varchar2(20)	See *Grad*	2
✓ *Dg_code1*	Varchar2(20)	Contains the ICD O Codes of the case	2
✓ *Dg_code2*	Varchar2(20)	Contains the ICD 10 Codes of the case	2
✗ *Pap*	Varchar2(4)	Contains the results of the PAP test detecting cancerous processes in the cervix.	2
✗ *Gyn_qualität*	Varchar2(10)	Gynecological description	2
✓ *Bereich*	Varchar2(1)	Contains the abbreviation of the department wherefrom the tissues were sent	2

Table 3.7: Description of the fields contained in all the tables of Imipatho. The tables are assigned to IDs, see table 3.6. The IDs are assigned to the fields of the DB. The content or parts of the content of the fields marked with the tick are migrated to the target DB.

3.2.4 MySQL database

The MySQL DB is called *grz_transfer* and contains one table called *grz_data*.

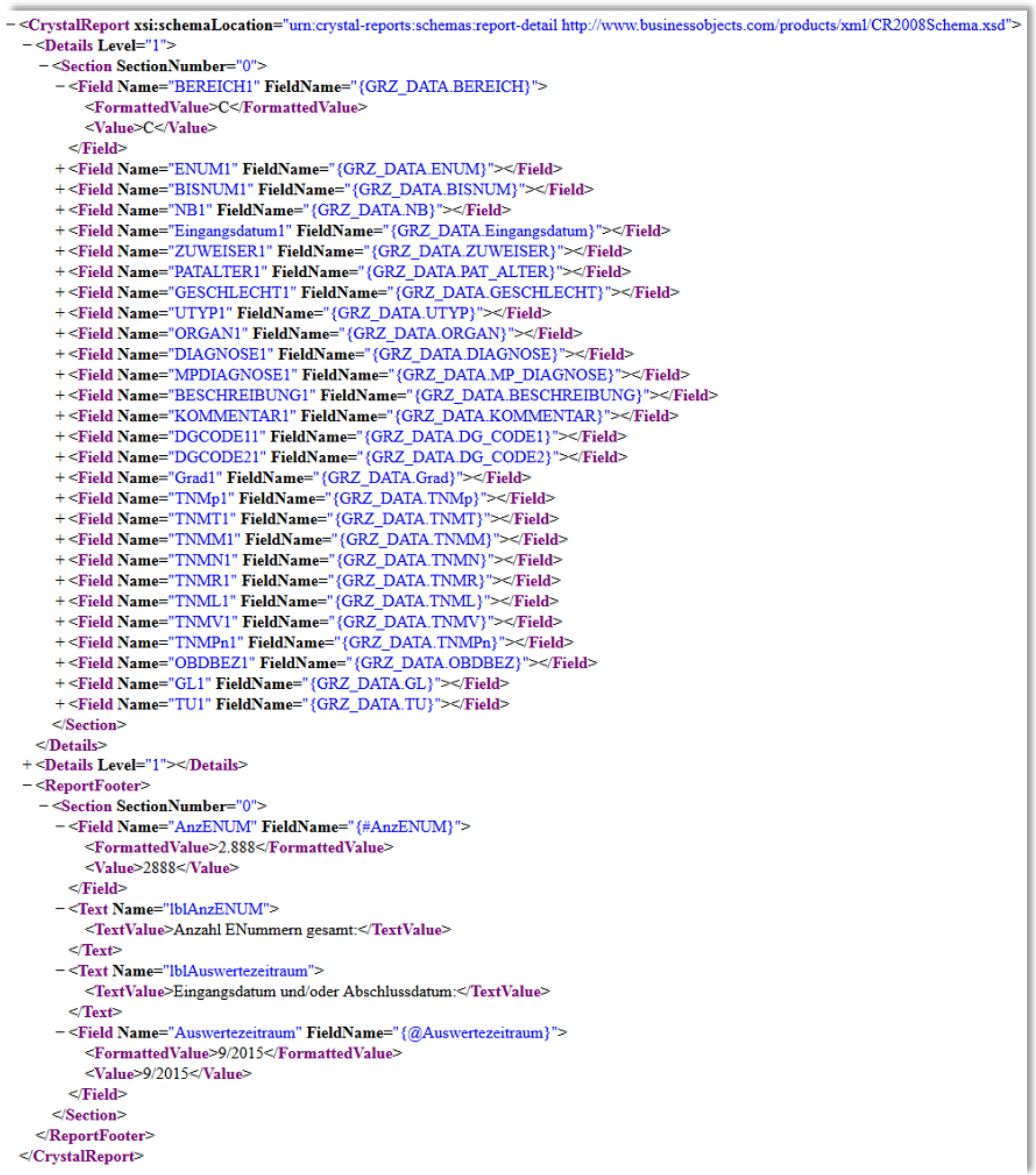
Field name	**Data type**	**Description**
✓ *Enum*	Varchar(10)	First number of the case, containing the year
✓ *Nb*	Varchar(2)	The supplement number of diagnostic finding
✓ *Bisnummer*	Varchar(2)	Last number of the case
✓ *Eingangsdatum*	Datetime	Contains the extraction date of the tissue
✓ *Pag_alter*	Varchar(3)	The age of the patient
✓ *Utyp*	Varchar(2)	Contains the abbreviation of the examination type, which describes the process of extraction
✓ *Geschlecht*	Varchar(1)	The sex of the patient
✓ *Organ*	Varchar(50)	Contains the organs whereof the tissues were extracted
✓ *Zuweiser*	Varchar2(10)	Contains information about the clinic respectively the doctor who sent the tissues
✓ *Diagnose*	Longtext	Contains the pathologic diagnosis of the pathologist
✓ *Mp_diagnose*	Longtext	Contains the molecular pathologic diagnosis of the pathologist
✓ *Beschreibung*	Longtext	The histologic result
✓ *Kommentar*	Longtext	Contains comments of the pathologist
✓ *GL*	Longtext	The underlying disease of the patient
✗ *Obdbez*	Varchar(30)	Autopsy designation
✓ *Tu*	Longtext	The coroner's inquest of the patient
✓ *Grad*	Varchar(30)	Parameter of the tumor staging
✓ *Tnmp*	Varchar(30)	See *Grad*
✓ *Tnmt*	Varchar(30)	See *Grad*
✓ *Tnmm*	Varchar(30)	See *Grad*
✓ *Tnmn*	Varchar(30)	See *Grad*
✓ *Tnmr*	Varchar(30)	See *Grad*

✓ *Tnml*	Varchar(30)	See *Grad*
✓ *Tnmv*	Varchar(30)	See *Grad*
✓ *Tnmpn*	Varchar(30)	See *Grad*
✓ *Dg_code1*	Varchar(30)	Contains the ICD O Codes of the case
✓ *Dg_code2*	Varchar(30)	Contains the ICD 10 Codes of the case
✓ *Bereich*	Varchar(1)	Contains the abbreviation of the department wherefrom the tissues were sent

Table 3.8: Description of the fields contained in the "grz_data" table of "grz_transfer". The content or parts of the content of the fields marked with the tick are migrated to the target DB.

3.2.5 XML files

Each month, the IMI transmits an XML file to the Biobank of Graz containing information about the stored tissues. Like the Biobank, the IMI is an institute of the Medical university of Graz. Several XML files exist, which have to be migrated to the source DB. Listing 3.1 presents the structure of the source XML files. The files contain equivalent data stored in the MySQL DB, see section 3.2.4.



```
- <CrystalReport xsi:schemaLocation="urn:crystal-reports:schemas:report-detail http://www.businessobjects.com/products/xml/CR2008Schema.xsd">
  - <Details Level="1">
    - <Section SectionNumber="0">
      - <Field Name="BEREICH1" FieldName="{GRZ_DATA.BEREICH}">
          <FormattedValue>C</FormattedValue>
          <Value>C</Value>
        </Field>
      + <Field Name="ENUM1" FieldName="{GRZ_DATA.ENUM}"></Field>
      + <Field Name="BISNUM1" FieldName="{GRZ_DATA.BISNUM}"></Field>
      + <Field Name="NB1" FieldName="{GRZ_DATA.NB}"></Field>
      + <Field Name="Eingangsdatum1" FieldName="{GRZ_DATA.Eingangsdatum}"></Field>
      + <Field Name="ZUWEISER1" FieldName="{GRZ_DATA.ZUWEISER}"></Field>
      + <Field Name="PATALTER1" FieldName="{GRZ_DATA.PAT_ALTER}"></Field>
      + <Field Name="GESCHLECHT1" FieldName="{GRZ_DATA.GESCHLECHT}"></Field>
      + <Field Name="UTYP1" FieldName="{GRZ_DATA.UTYP}"></Field>
      + <Field Name="ORGAN1" FieldName="{GRZ_DATA.ORGAN}"></Field>
      + <Field Name="DIAGNOSE1" FieldName="{GRZ_DATA.DIAGNOSE}"></Field>
      + <Field Name="MPDIAGNOSE1" FieldName="{GRZ_DATA.MP_DIAGNOSE}"></Field>
      + <Field Name="BESCHREIBUNG1" FieldName="{GRZ_DATA.BESCHREIBUNG}"></Field>
      + <Field Name="KOMMENTAR1" FieldName="{GRZ_DATA.KOMMENTAR}"></Field>
      + <Field Name="DGCODE11" FieldName="{GRZ_DATA.DG_CODE1}"></Field>
      + <Field Name="DGCODE21" FieldName="{GRZ_DATA.DG_CODE2}"></Field>
      + <Field Name="Grad1" FieldName="{GRZ_DATA.Grad}"></Field>
      + <Field Name="TNMp1" FieldName="{GRZ_DATA.TNMp}"></Field>
      + <Field Name="TNMT1" FieldName="{GRZ_DATA.TNMT}"></Field>
      + <Field Name="TNMM1" FieldName="{GRZ_DATA.TNMM}"></Field>
      + <Field Name="TNMN1" FieldName="{GRZ_DATA.TNMN}"></Field>
      + <Field Name="TNMR1" FieldName="{GRZ_DATA.TNMR}"></Field>
      + <Field Name="TNML1" FieldName="{GRZ_DATA.TNML}"></Field>
      + <Field Name="TNMV1" FieldName="{GRZ_DATA.TNMV}"></Field>
      + <Field Name="TNMPn1" FieldName="{GRZ_DATA.TNMPn}"></Field>
      + <Field Name="OBDBEZ1" FieldName="{GRZ_DATA.OBDBEZ}"></Field>
      + <Field Name="GL1" FieldName="{GRZ_DATA.GL}"></Field>
      + <Field Name="TU1" FieldName="{GRZ_DATA.TU}"></Field>
      </Section>
    </Details>
  + <Details Level="1"></Details>
  - <ReportFooter>
    - <Section SectionNumber="0">
      - <Field Name="AnzENUM" FieldName="{#AnzENUM}">
          <FormattedValue>2.888</FormattedValue>
          <Value>2888</Value>
        </Field>
      - <Text Name="lblAnzENUM">
          <TextValue>Anzahl ENummern gesamt:</TextValue>
        </Text>
      - <Text Name="lblAuswertezeitraum">
          <TextValue>Eingangsdatum und/oder Abschlussdatum:</TextValue>
        </Text>
      - <Field Name="Auswertezeitraum" FieldName="{@Auswertezeitraum}">
          <FormattedValue>9/2015</FormattedValue>
          <Value>9/2015</Value>
        </Field>
      </Section>
    </ReportFooter>
  </CrystalReport>
```

Listing 3.1: Contents of the XML files. Each dataset is accessible by opening the "details" node. The details node comprises the "Section" node, which contains the children called "Fields". These fields contain the important data and the structure is equivalent to the structure in the MySQL source DB, see section 3.2.4.

3.2.6 Creation of aliases of the sources to be migrated

The source DBs contain German table and field names. In this section aliases are introduced for all the tables and fields of the source DBs and XML files that are migrated to gain easier access to the content.

Table 3.9 illustrates the fields and their created aliases that are used instead of their original names in this thesis. For better recognition the aliases are written in italic style.

Alias / ID	**Imi_grz_data**	**Imi_aura_data**	**Tissues**	**XML files / MySQL DB**
AuraId	❌	❌	Aura_idx	❌
FirstNo	Befundnummer	Befundnummer	Unum	Enum
SupplementNo	Nb	Nb	Last character of AuraId	Nb
LastNo	Bisnum	Bisnum	Calculable: FirstNo + NumFindings - 1	Bisnum
Department	Bereich	❌	❌	Bereich
Sender	Einsender	Einsender	Einsender	Zuweiser
ExaminationType	Unttyp	Unttyp	Utyp	Utyp
ExaminationYear	Untjahr	Untjahr	Ujahr	The year of *ExaminationDate*
ExaminationDate	❌	❌	Udate	Eingangsdatum
PatientAge	Pat_age	Pat_age	Calculable from ExaminationDate and PatientBirth	Pat_alter
PatientSex	Pat_sex	Pat_sex	❌	Gechlecht
Forename	❌	❌	Vorname	❌
Surname	❌	❌	Nachname	❌
DateOfBirth	❌	❌	Geburtsdatum	❌
Organ	Material	Material	Organ	Material
Diagnose	Diagnose	Diagnose	Diagnose	Diagnose
DiagnoseMolecular	❌	❌	❌	Mpdiagnose
HistologicResult	Histo	Histo	❌	Beschreibung
MacroscopicResult	Makro	Makro	❌	❌
Comment	❌	❌	❌	Kommentar
IcdO	Dg_code1	❌	❌	Dgcode1
Icd10	Dg_code2	❌	❌	Dgcode2
UnderlyingDisease	Grundleiden	Grundleiden	❌	Gl
AdditionalDisease	Nebenleiden	Nebenleiden	❌	❌

CoronersInquest	Todesursache	Todesursache	❌	Tu
BooleanOrganFlags	❌	❌	❌	All Boolean categorized organs, see section 0
TnmParameters	All parameters of the tumor staging, see section 0	❌	❌	All parameters of the tumor staging, see section 3.2.4
HER2Parameters	❌	❌	All HER2 parameters, see section 0	❌

Table 3.9: Overview of the contents of the source tables by creating aliases: The structure of the tables is similar, but the names of the columns differ. The aliases respictively IDs are introduced for easier identifictaion of the fields in the sources.

3.3 Investigation of the sources

This section gives an overview of the contents of the source DBs and files. The source information, which has to be migrated is located in four DBs and five XML files, see table 3.10 containing information about the tissues stored in the Biobank Graz.

Source	**Number of contained tables / files**	**Number of tables /files relevant for migration**
Microsoft Access DB	79	74
Oracle 10g DB 1	2	2
Oracle 10g DB 2	2	2
MySQL DB	1	1
XML files	5	5

Table 3.10: Overview of the sources and their contained objects

3.3.1 Microsoft Access DB

Table 3.11 to illustrates the relevant objects for the migration process of the MSA DB. The number of contained datasets and information about the primary key are illustrated. The majority of the datasets is contained within one table storing about 1.3 million unique datasets. All sources within DBs have in common that they are not related to other tables. Only two of 79 tables contain a primary key, therefore only datasets in these two tables can be clearly distinguished. The remaining tables can contain duplicated data.

Source table	**Number of datasets**	**Primary key**	**Fields of the primary key**	**Contained examination years**
ALLE METASTASE	47.113	✗		1984 – 2006
CA+Colotis	153	✗		1984 – 2006
Colitis Ulcerosa	3.415	✗		1984 – 2006
COLITIS+Patienten	1.300	✗		1984 – 2006
KOLON CA DIE META:HA-BEN	13.100	✗		1984 – 2006
KOLON CA_NEU	10.339	✗		1984 – 2006
KOLON META OHNE DUP-LIKATEN	5.630	✗		1984 – 2006
Mamma metastase nach 2006 + Primär CA	2.182	✗		1984 – 2006
METASTASEN VON CRC PATIENTEN	5.473	✗		1984 – 2006
Patienten Harnblase-Prim+Metastase	233	✗		1984 – 2005
2001-2004	2.711	✗		2001 – 2004
2003-2005	1.117	✗		2003 – 2005
Arthritis	855	✗		1984 – 2006
Astrozytom	99	✗		1986 – 2006
Cholangiozellu	50	✗		1986 – 2005
CRC 2001-2004	710	✗		2001 – 2004
crc in leber	268	✗		1984 – 2006
CRC Lebermeta	1.977	✗		1984 – 2006
Dermatomyositis	55	✗		1988 – 2006
Endometrium CA	147	✗		2002
Fibröse Dysplasie	139	✗		1984 – 2006
*FISH ******	136	✗		1985 – 2006

FISH 2005-2006	143	✗		2005, 2006
GIST	91	✗		1984 – 2006
Glioblast	1.551	✗		1984 – 2006
Hepatocelll	199	✗		1984 – 2006
HER 2 NEU (3+)	2.484	✗		1984 – 2006
her2	961	✗		1989 – 2006
Hirn	662	✗		1984 – 2006
Histozytom	2.598	✗		1984 – 2006
Hoden	44	✗		1984 – 1998
Keinklein	408	✗		1984 – 2006
leber biopsie	1.773	✗		1984 – 2006
LIPOSARKOM 2003	10	✗		2003
LUNGEN CA ALLE SEB	715	✗		1996 – 2006
Malignes Melanom	2.599	✗		1984 – 2006
mamma 200	515	✗		1987 – 2006
Mamma ab 2000 alles DG	12.653	✗		1984 – 2006
Mamma CA+Meta HER2/Neu	1.516	✗		1984 – 2006
Mamma meta	18.530	✗		1984 – 2006
Mamma meta 2001-2002	603	✗		2001 – 2002
Mamma meta 2001-2002 tubul	1,000	✗		2001 – 2002
mamma normal gewebe	161	✗		1984 – 2006
MammaCa	13.111	✗		1984 – 2006
medulloblastom	24	✗		1994 – 2003
Metastase Harnblase CA	133	✗		1984 – 2006
Mitellhirn	21	✗		1993 – 2006
ohneadenocervix	13	✗		1995 – 2004
, onko erweiterung	33	✗		1985 – 2005
onko2	159	✗		1984 – 2006

Onkozyt	110	✗		1984 – 2006
Onkozytom	69	✗		1985 – 2005
Ovarial CA	690	✗		1984 – 2006
Pankreas	1.301	✗		1984 – 2006
Plasmozytom	2.409	✗		1984 – 2006
Pleomorphe liposarkom	39	✗		1984 – 2005
Prostata mach	147	✗		1984 – 2005
ProstataCA	93	✗		2006
Rhabdomyosarkom	3.403	✗		1984 – 2006
Rückenmark	18	✗		1993 – 2005
score 0 und 1	704	✗		1991 – 2005
score 2	68	✗		1992 – 2005
score3	114	✗		1989 – 2005
sentinel11	80	✗		1989 – 2006
Wilms Tumor	64	✗		1984 – 2006
Nase	6.682	✗		1984 – 2006
Nasenpolype	21	✗		2002 – 2005
NSCLC	1.517	✗		1984 – 2006
mamma ca	13.111	✗		1984 – 2006
HCC aus 2006	13	✗		2006
FISH Ausgewählt	19	🔑	*AuraId*	2005 – 2006
BefundeAnn2006_2009	171.932	✗		2006 – 2009
Organe 01/01/1984 - 30/06/2006	1.294.119	🔑	*AuraId*	1984 – 2006

Table 3.11: Information about the relevant objects within the MSA DB: All objects relevant for the migration process are listed. The tables cover the years from 1984 to 2009.

3.3.2 Oracle 10g database 1

Table 3.12 shows the tables within the Oracle DB 1. The DB contains two tables. Table *Fall_data* contains about 1.3 million datasets. The size of the second table, *Befunde20062009* comprises about 170.000 datasets. The tables are not linked to each other. Both of the tables contain a primary key which guarantees unique datasets.

Source table	**Number of datasets**	**Primary key**	**Fields of the primary key**	**Contained examination years**
Befunde20062009	171.932	🔑	*FirstNo, SupplementNo*	2006 - 2009
Fall_data	1.294.081	🔑	*AuraId*	1984 - 2006

Table 3.12: Information about the relevant objects within the Oracle 10g database 1: All objects relevant for the migration process are listed. The tables cover the years from 1984 to 2009.

3.3.3 Oracle 10g database 2

The second Oracle DB contains two tables, which are both relevant for the migration, see table 3.13. The tables cover the years from 1984 to 2009 and contain both a primary key, guaranteeing unique datasets. The larger table contains 911.780 datasets and the smaller one about 187.651. The tables are not related to each other nor they are related to other sources.

Source table	**Number of datasets**	**Primary key**	**Fields of the primary key**	**Contained examination years**
Imi_grz_data	187.651	🔑	*FirstNo, SupplementNo*	2005 - 2009
Imi_aura_data	911.780	🔑	*FirstNo, SupplementNo*	1984 - 2005

Table 3.13: Information about the relevant objects within the Oracle 10g database 2. All objects relevant for the migration process are listed. The tables cover the years from 1984 to 2009.

3.3.4 MySQL database

The MySQL DB contains one table, which is not related to any other table. It contains data from 2009 to 2015 and all contained datasets are unique because of the assigned primary key, see table 3.14.

Source table	**Number of datasets**	**Primary key**	**Fields of the primary key**	**Contained examination years**
Grz_data	185.476	🔑	*FirstNo, SupplementNo, Department*	2009 – 2015

Table 3.14: Information about the relevant object within the MySQL database. All objects relevant for the migration process are listed. The table covers the years from 2009 to 2015.

3.3.5 XML files

Table 3.15 illustrates the number of datasets and the main examination year of each XML source file. The names of the sources give information about the contained information, year and month. Additionally, the sources can contain information about HCIs of previous months. In this case the information contained in the newer XML file is the updated and right one and makes the datasets of previous XML files obsolete.

Source file	**Number of datasets**	**Contained examination years**
BBmonthlyUpdate_2015_09.xml	2.888	2015
BBmonthlyUpdate_2015_10.xml	3.104	2015
BBmonthlyUpdate_2015_11.xml	3.159	2015
BBmonthlyUpdate_2015_12.xml	2.849	2015

Table 3.15: Information about the content of the XML files

3.4 Design and implementation of the Oracle 11g target database

The sources are migrated to an Oracle 11g target DB. The target DB does not and is designed and implemented as part of this work. All the marked information about the sources has to be stored in the target DB, see section 3.1. Knowing which data has to be stored and what the properties of the data are an Oracle 11g DB is designed and implemented, see figure 3.2. The target DB contains 14 tables. The first column of each table is the primary key, recognizable by the prefix “pk”. Foreign keys have the prefix “fk”. The important tables are explained in more detail:

Histocircles: This table stores all the information about the HCI of the FFPE tissue sample. The *FirstNo* is split up into the not nullable fields *histo_year* and *from_no* representing the lowest HNO associated with the HCI. The combination of these two fields is declared as unique to ensure that multiple insertions of same *FirstNos* are not possible. The departments, senders and patients are stored within the table by the use of the foreign keys of the corresponding lookup tables *lookup_departments*, *lookup_patients* and *lookup_senders*.

Medical_documentations:	This table contains the information about the HNO. The table is linked to the table *Histocircles*. The foreign key of the HCI and the column *supplement_no* form a unique constraint together. Additionally, the table is linked to the tables *lookup_sources* and *lookup_examination_types* to store the examination type of the sample and the source of the dataset.
Lookup_parameters:	The parameter types for tumor staging and HER2 are stored within this table. Thus, each HNO can contain various number of parameter values. An additional link table *link_params_to_med_docus* is implemented holding information about the parameter type, the HNO and the value of the parameter.
Lookup_organs:	This table contains the organs used in the *BooleanOrganFlags*. Twenty different organs are inserted into this lookup table. The HCIs are linked to this table by the use of the table *link_organs_to_histos_bool* storing information about the organ and the corresponding HNO.
Lookup_keywords:	Frequently used keywords for searching in DBs of Biobanks are stored within this table. This dictionary contains about 1.000 keywords. Additionally, various spellings of the keywords are stored in the field pattern by the use of REGEXs. HNOs can be linked for faster queries with *lookup_keywords* by the use of table *link_meddoc_to_keyword* containing the key of the HNO and the key of the keyword.

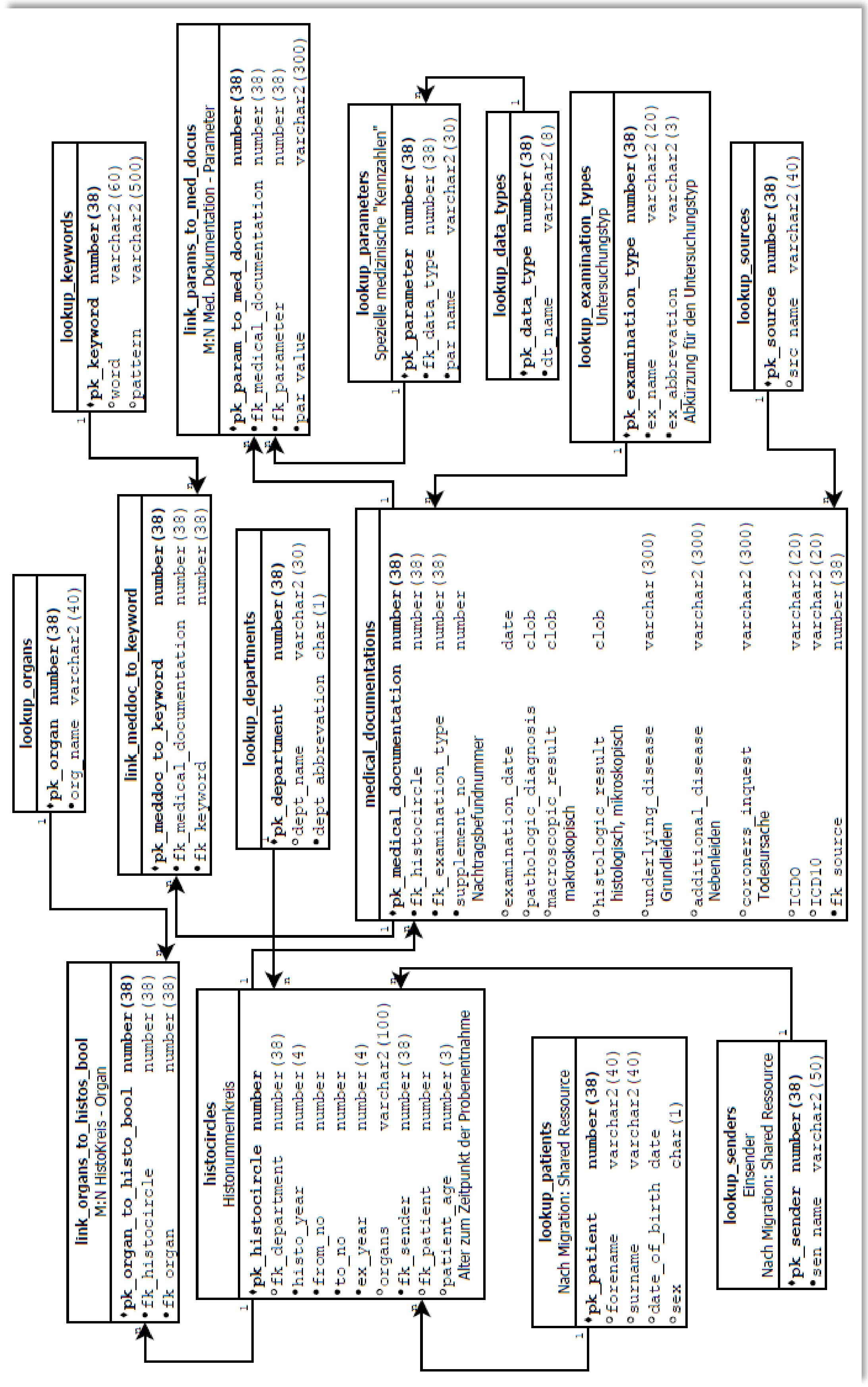


Figure 3.2: Entity relationship model of the target Oracle 11g database

3.5 Implementation of a generic data migration process

This explanation of the data migration includes a detailed example for each step of the process. Figure 3.3 illustrates the phases of the migration process.

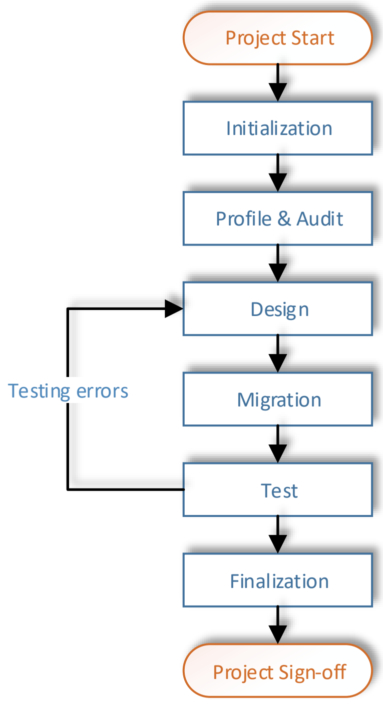


Figure 3.3: Migration process in this thesis: The first phase of the migration process is the initialization, followed by the profile and audit phase, followed by the design phase, followed by the migration phase, followed by the test phase and followed by the finalization phase. After the finalization the project is ready to be signed off.

3.5.1 Initialization phase

After the acknowledgement that the migration process can start within the organization, the data migration platform has to be prepared. The migration platform is used for three main purposes:

1. On this machine the migration script is developed.
2. The tests are performed on this platform.
3. The migration runs on this system.

The hardware and the OS of the platform has to be defined and installed. Additionally, the software applications for development have to be installed. Furthermore, the software has to be configured to gain access to the sources, which have to be migrated. The initializing phase comprises an analytical part. Based on the result of the analysis the appropriate migration strategy is selected. Table 3.16 gives an overview of the installed hard- and software of the migration platform.

Processor	Intel® Core™ i7-3770 CPU 3.40 GHz
Random access memory (RAM)	8 Gigabytes
OS	Windows 7 Enterprise
Client for accessing the DBs	MySQL Workbench/J
IDE for software development	Eclipse IDE
Environments	Java SE Development Kit 1.8
DB software	Microsoft Access 2013
Drivers	MySQL Connector/ODBC Version 5.3 Oracle Database 11g Release 2 JDBC drivers
Additional software	Dia diagram editor, Notepad++ MindFusion XML viewer Git TortoiseGit

Table 3.16: Overview of the hard- and software of the migration platform

The MSA DB and the XML sources are copied on the migration platform. The rest of the DBs are accessed via the specified DB client, see table 3.16. Thus, the JDBC drivers have to be configured in the client software.

The outcome of the analysis of the initialization phase is that the MSA DB is the productive one and the rest of the DBs are not be used. Because an Access DB can be easily duplicated by copying the MSA DB file and the other DBs are not running productive the migration strategy of this thesis is a Big-bang approach.

3.5.2 Profile and audit phase

In order to process this phase, the sources have to be identified. Usually the sources are known before the start of a migration project. During working on this thesis the content of the sources was identified, see section 3.1.

The profile and audit phase are about the understanding of the data content and the identification of the data which has to be effectively migrated. Nevertheless, some of the sources could be excluded during the identification of the sources, see section 3.2. Data analyses help to identify relevant data which has to be migrated. Additionally, the analysis helps with the cleansing of the data by potentially uncovering redundancies and inconsistencies. The data cleansing itself may resolve parts of these issues. Issues, which cannot be resolved by applying the default rules defined have to be checked manually.

With the knowledge gained through the identification of the sources, see section 3.1, the analysis and cleansing of each of the sources is executable. The management decides which data of the sources has to be migrated. Data marked in green is assigned for migration, see section 3.1. and then goes through data cleansing. The data cleansing comprises the identification of redundancies and inconsistencies in this thesis. The data cleansing is executed in SQL. In the source tables additional columns containing information about consistency are added. If a dataset contains inconsistencies, the corresponding consistency column is marked. If the dataset is consistent, which means that no consistency column contains an entry the dataset is ready to be migrated to the target Oracle 11g DB. The additional consistency and migration columns for each of the source tables are explained in detail, see section 3.5.2.1 – section 3.5.2.5.

First analysis shows that the MSA source contains inter alia lots of tables with similar structure. The Oracle DB 1 and Oracle DB 2 contain two tables similar to the MSA source. All of the mentioned data sources contain data from the beginning of 1984 to the middle of 2009. Because of the great similarity between the MSA and the Oracle 10g DBs they are compared and redundancies are identified in first place. The MySQL source contains data from 2009 to the middle of 2015 and does not overlap with data from the other sources. Since one XML source contains the data of one month from the second half of 2015, cleansing of these sources is executed during the migration process, because every month - even after the finalization of this thesis - new monthly data of the same type has to be migrated to the target DB. With this analysis the following data cleansing steps can be concluded:

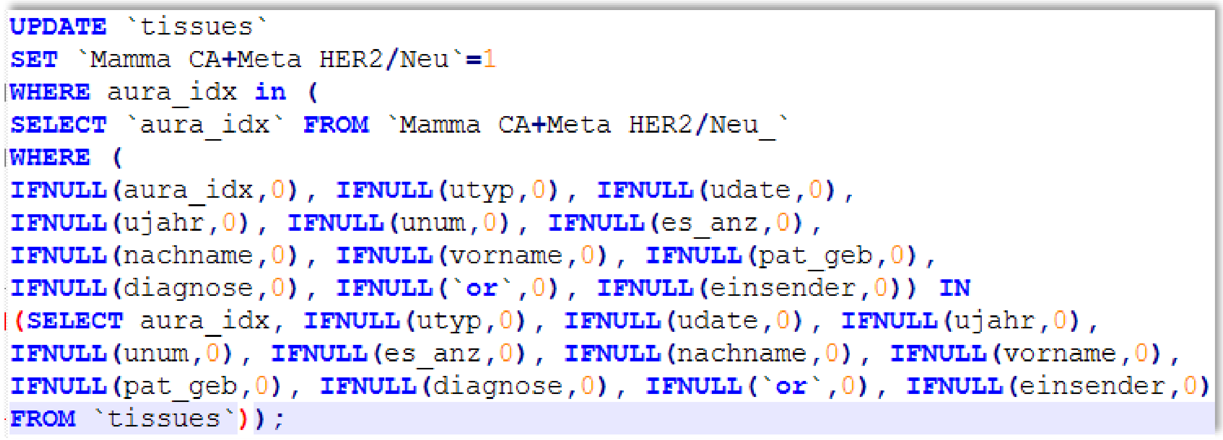
1. Identification of redundancies of the data contained in the tables of the MSA source, see section 3.5.2.1
2. Resultant data of step 1 is compared and cleansed with Oracle DB 1, see section 3.5.2.2
3. Cleansing of the result of step 2, see section 3.5.2.3
4. Cleansing of the Oracle DB 2, see section 3.5.2.4

5. Resultant data of step 3 is compared and cleansed with the resultant data of step 4, see section 3.5.2.5
6. Cleansing of the MySQL DB, see section 3.5.2.6
7. Cleansing of the XML files is done during the migration of each file, see section 3.5.2.7

3.5.2.1 Data cleansing between the tables of the Microsoft Access database

The relevant MSA tables are exported to MySQL for data cleansing. MSA offers the possibility to export DB objects e.g. tables to MySQL by the use of the ODBC interface. The grouped tables with the ID from one to five contain the same fields, see section 0. The tables with group ID number six contain the same fields and additionally four fields not contained in the rest of the tables. The tables contained in the groups from one to six are compared for duplicate data. The steps for the comparison are:

1. A copy of the main table is created and named *tissues*.
2. The new *tissues* table is extended by a Boolean field for each compared table. The name of the Boolean field gets the name of the compared table.
3. New tables are created for each table not containing the *AuraId* column by selecting all the distinct datasets of the tables. The name of the new tables is simply the old one extended by the prefix "_".
4. Datasets in the tables containing the *AuraId* and the datasets of the newly generated tables are compared directly with the dataset of the main *tissues* table. The corresponding Boolean field of the *tissues* table of a dataset is set to true, if the *tissues* table contains the equivalent dataset of the compared table by the use of SQL update scripts, see listing 3.2.
5. After the first four steps of cleansing the results are investigated.



```
UPDATE `tissues`
SET `Mamma CA+Meta HER2/Neu`=1
WHERE aura_idx in (
SELECT `aura_idx` FROM `Mamma CA+Meta HER2/Neu_`
WHERE (
IFNULL(aura_idx,0), IFNULL(utyp,0), IFNULL(udate,0),
IFNULL(ujahr,0), IFNULL(unum,0), IFNULL(es_anz,0),
IFNULL(nachname,0), IFNULL(vorname,0), IFNULL(pat_geb,0),
IFNULL(diagnose,0), IFNULL(`or`,0), IFNULL(einsender,0)) IN
(SELECT aura_idx, IFNULL(utyp,0), IFNULL(udate,0), IFNULL(ujahr,0),
IFNULL(unum,0), IFNULL(es_anz,0), IFNULL(nachname,0), IFNULL(vorname,0),
IFNULL(pat_geb,0), IFNULL(diagnose,0), IFNULL(`or`,0), IFNULL(einsender,0)
FROM `tissues`));
```

Listing 3.2: SQL example update statement of datasets contained in the main table: This statement sets the Boolean field "Mamma CA+Meta HER2/Neu" of the main table to true, if the equal dataset is contained in the corresponding table.

The result of the analysis in Step 5 points out, that only a few datasets are not contained in the main table. After further analyses it becomes apparent that all the datasets of the compared tables are contained within the *tissues* table. The smaller tables contained datasets without information or cut information. The result of the cleansing is, that all the information stored in the MSA DB can be reduced to two tables: *tissues* and *findings*, which is an exact copy of *BefundeAnn2006_2009*. The additional information about the *Her2Parameters* contained in the table "*FISH Ausgewählt*" is copied to *tissues* by adding and copying the data into the four new created *Her2Parameters* columns. The result of this cleansing phase is illustrated in figure 3.4.

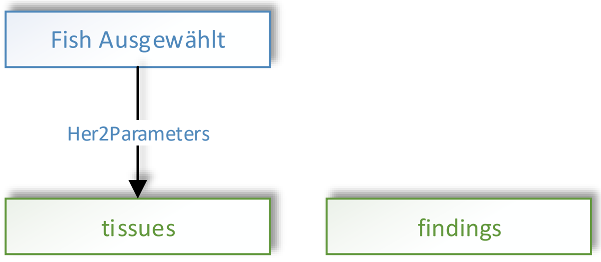


Figure 3.4: Result of the cleansed data in the Microsoft Access database: All the information about the MSA DB is stored in the tables "tissues" and "findings". The Her2Parameters of the table "Fish Ausgewählt" are added to the "tissues" table. The rest of the tables is either contained in one of the two mentioned tables or not important for the migration.

3.5.2.2 Data cleansing between the cleansed data of 3.5.2.1 and Oracle database 1

After the cleansing of the MSA DB, see section 3.5.2.1 the cleansed data is exported to the Oracle 10g DB called *Imirep*, see section 3.2.2. The two tables, *tissues* and *findings*, holding all the information about the MSA DB are exported from MySQL to the *Imirep* DB by the use of the built in *DataPumper* of *SQL Workbench/J*, see section 2.2.7.

Section 0 and section 3.2.2 show, that the structure of the tables *findings* and the *Befunde20062009* of *Imirep* are similar. The same applies for the tables *tissues* and the *fall_data* table of *Imirep*. Because of these similarities two comparisons are made:

- Comparison 1: *findings* with *Befunde20062009*
- Comparison 2: *tissues* with *fall_data*

The tables are compared to each other by applying the following steps:

1. The number of datasets in each of the tables is checked.
2. Columns for enabling an SQL join between the tables are selected.
3. By the use of the SQL join, the datasets of the tables are compared with each other. The equivalent columns of the tables, see section 3.2.6 are compared. Possible NULL values are compared with each other by replacing integer NULL values with a zero, empty Strings with the string *"0"* and empty dates with *"01.01.1950"*. This date is possible because tissues are available from 1984 to 2015, see section 3.3.
4. In the case of differences between the datasets of the compared tables an analysis is made if the comparison is revised or abandoned.
5. Implementation and execution of the revised comparison.
6. Analysis of the result of the revised comparison. If the comparison is not successful, the datasets are inconsistent and have to be marked for manual inspection.
7. Analysis of the datasets, which cannot be joined.

The result of the analysis in step 4 shows, that the contents of the compared fields do not contain the same character sets. To achieve a better comparison between the data special characters like umlauts of the German language are replaced. Table 3.17 and table 3.18 show the replacements for the revised comparisons in detail.

Alias	**Replaced characters**	**Replaced by**
Organ	Leading and trailing whitespaces, (,), “	Empty string
UnderlyingDisease, CoronersInquest, AdditionalDisease, HistologicResult, MacroscopicResult	Leading and trailing whitespaces, ¿, \\n,“, carriage return, line feed, horizontal tab	Empty string
	All characters except alphanumeric, numeric characters, and whitespaces.	Empty string
Rest of the string IDs	Leading and trailing whitespaces	Empty string

Table 3.17: Adaptions for the revised comparison 1: The two compared tables findings and befunde20062009 contain different character sets. To be able to compare the content of the two tables characters have to be replaced in the fields.

After the revision of comparison 1 only a handful of datasets are not equal. A manual check shows that the *HistologicResult* and *MacroscopicResult* of *befunde20062009* contain truncated data and the corresponding datasets in *findings* contains the non-truncated datasets. One dataset in each of the tables cannot be joined. The result of the manual revision is the datasets contain the same information with different encoding. Thus, all of the information stored in table *befunde20062009* is stored in table *findings*.

Alias	**Replaced characters**	**Replaced by**
Forename, Surname	All characters, except alphanumeric characters	Empty string
Sender, Organ, Diagnose	All characters except alphanumeric and numeric characters. Additionally, alphanumeric characters are converted to uppercase. Umlauts get a special replacement: the corresponding Latin character followed by an 'E'.	Empty string, Ä→ AE Ö→ OE Ü→ UE

Table 3.18: Adaptions for the revised comparison 2: The two compared tables "findings" and "befunde20062009" contain different character sets. To be able to compare the content of the two tables characters have to be replaced in the fields.

After the revision of comparison 2, 254 datasets do not match. These datasets are revised manually. The manual revision shows, that inadequate characters such as *È*, *Î*, etc. are contained in the diagnose of the *tissues* table. These characters are ignored by the next SLQ comparison. After the adaption additional 253 matches are achieved. One dataset in the *fall_data* table does not contain a sender. The corresponding dataset of the tissues table does. In this case the dataset of the tissues table is the valid one containing additional information.

	Comparison 1: findings - befunde20062009	**Comparison 2: tissues - fall_data**
Number of datasets table 1	171.932	1.294.119
Number of datasets table 2	171.932	1.294.081
Primary key table 1	*FirstNo, SupplementNo*	*AuraId*
Primary key table 2	*FirstNo, SupplementNo*	*AuraId*
Number of joined datasets	171.931	1.294.081
Matched datasets	12.935	103.271
Matched datasets after revision	171.793	1.293.827

Final result after manual revision	171.932	1.294.081
100 % of the datasets in one of the compared tables	✓	✓

Table 3.19:Results of the comparison between Imirep DB and MSA DB: The comparison of findings and befunde20062009 shows that they contain equal datasets. The primary key including "FirstNo" and "SupplementNo" is equal. After step three of the comparison about 7,5 percent of the contained datasets are equal. After the revision of the comparison less than one percent of the datasets are not equal. After the manual check of the not matched and not joined datasets, it can be said that table befunde20062009 is fully contained in the findings table. Comparison 2 shows a difference in the number of datasets in the tables. The primary key in both of the tables is the "AuraId" column. 103.271 datasets are equal in the simple comparison. After the refinement, 254 datasets are not equal. After the manual revision it is figured out, that characters not needed such as î, é, etc. are included in the diagnose. Table fall_data is contained in table tissues without these characters.

Similar to section 3.5.2.1, all the data is contained in the original MSA tables *findings* and *tissues*, see table 3.19 and figure 3.5.

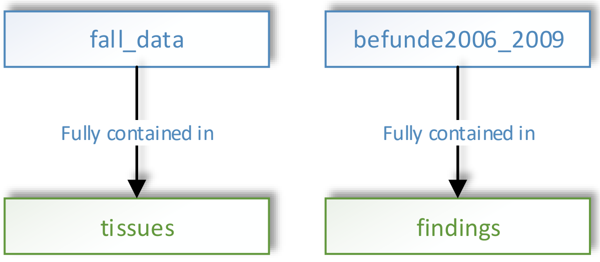


Figure 3.5: Result of the cleansed data in this cleansing phase: All the information is stored in the tables "tissues" and "findings". Table "fall_data" is contained in table "tissues" and table "befunde2006_2009" is contained in table findings.

3.5.2.3 Data cleansing within the remaining data of 3.5.2.2

Table *tissues* is treated first: The *FirstNo* of this table contains its information in different formats. For an easier comparison between the datasets a new column *No* is added, where the information about *FirstNo* is stored in a single eight-digit string format, see table 3.21. Another new column *Nb* is added to the table. The one-digit *SupplementNo* is extracted out of the *AuraId* and stored in the new column. Additionally, the default migration flag column is added. Table 3.20 gives an overview of the new columns and its meanings.

Column	**Explanation**
No	Contains the *FirstNo* in a standardized eight-digit format.
Nb	Contains the one digit *SupplementNo* extracted out of the last position of the *AuraId*.
Migration	Inconsistent datasets are marked with an "X" meaning that the dataset is not migrated.

Table 3.20: Overview and explanation of the new introduced columns: For an easier cleansing procedure, the "FirstNo" and the "SupplementNo" are stored in newly introduced columns in a standardized format. A new migration flag column is introduced to mark inconsistent data.

The new primary key with *No* and *Nb* is set. If it is not possible to apply the new key the causing datasets are marked to be not migrated. The causing datasets are identified by querying the DB for duplicated contents in the column *No*. With the introduced *No* a new temporary table containing only the distinct datasets of a HCI, see section 3.1, is created. HCIs, which occur more than one time in the new temporary table are marked by adding an 'X' in the migration flag column in the *tissues* table as to be not migrated. Datasets, which do not contain a value in *No* are marked as well and are not migrated.

***FirstNo* source format**	**Additional columns**	**Example**	**New format example**
One digit	Examination year	1, 2000	00000001
Two digits	Examination year	12, 2000	00000012
Three digits	Examination year	123, 2000	00000123
Four digits	Examination year	1234, 1990	90001234
Five digits	Examination year	12345, 2000	00012345
Six digits	Examination year	123456, 2000	00123456
Seven digits		1123456	01123456
Eight digits		88123456	88123456
Ten digits		2005123456	05123456

Table 3.21: "FirstNo" conversion of table tissues: The table shows the nine different formats of the "FirstNo" column contained in the tissues table. The new "No" column contains the new standardized format. To achieve the target format additional columns of the source table are eventually needed.

Table *findings* shows a similar structure like the *imi_grz_data* table contained in Oracle DB 2 and is compared and cleansed later, see section 3.5.2.5. Figure 3.6 illustrates the process of this cleansing phase.

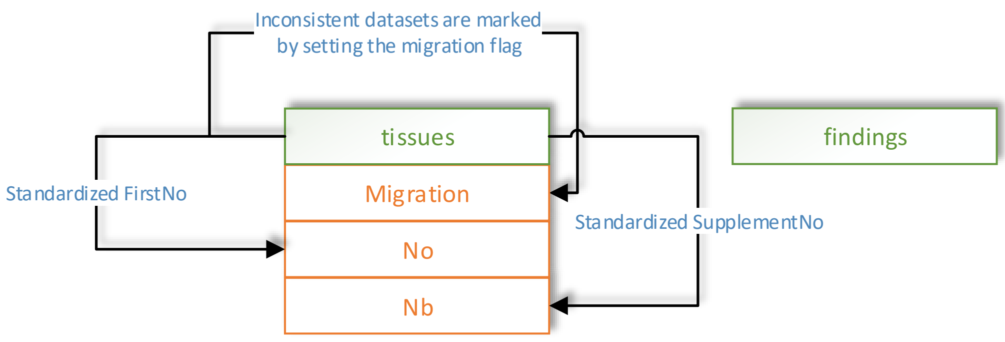

Inconsistent datasets are marked by setting the migration flag
tissues
Migration
No
Nb
findings
Standardized FirstNo
Standardized SupplementNo

Figure 3.6: Result of the cleansed data in this cleansing phase: All the information is stored in the tables "tissues" and "findings". The columns "No", "Nb" and "Migration" are added to table "tissues" where standardized information about "FirstNo" and the "SupplementNo" are stored. With the newly introduced standardized columns, the HCIs contained in the table are compared. Inconsistent datasets are marked with an "X" in the migration flag column.

3.5.2.4 Data cleansing within Oracle database 2

The Oracle DB 2 contains two different tables. The two tables, *Imi_grz_data* and *Imi_aura_data*, are extended by the migration flag column. Similar to the cleansing of table *tissues*, see section 3.5.2.3, new temporary tables with the distinct information about the HCIs of the tables are created. *FirstNos* contained more than one time are marked in original tables by writing an "X" into the migration flag column. Figure 3.7 illustrates the cleansing of the Oracle DB 2.

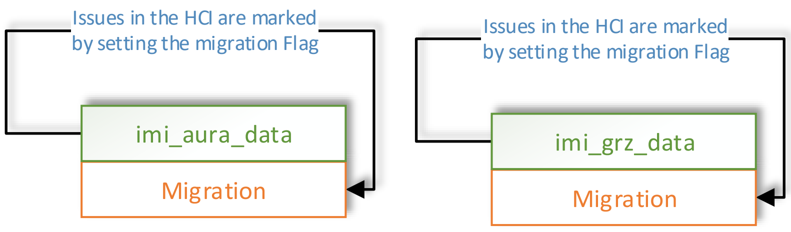

Issues in the HCI are marked by setting the migration Flag
imi_aura_data
Migration
Issues in the HCI are marked by setting the migration Flag
imi_grz_data
Migration

Figure 3.7: Result of the cleansed data in this cleansing phase: Oracle 11g DB 2 contains the tables "imi_aura_data" and imi_grz_data. The column "Migration" is added to the tables. The HCIs contained in the tables are compared and tested for inconsistencies. Datasets containing issues are marked with an "X" in the migration flag column.

3.5.2.5 Cleansing the remaining data of 3.5.2.3 and 3.5.2.4

The remaining data consists of the tables *tissues* and *imi_aura_data*, which are extended inter alia by the migration flag column. The tables *findings* and *imi_grz_data* have to be compared and cleansed as well. *Tissues* and *findings* are imported into the *Imipatho* Oracle 10g DB by the use of the *DataPumper* available in *SQL workbench/J*, see section 2.2.7.

Section 0 and section 0 show, that the structure of the imported *findings* and the Oracle 10g table *imi_grz_data* is similar. Thus, a comparison between the two tables is made. The comparison algorithm of data cleansing between the tables of MSA and Oracle DB 1, see section 3.5.2.2 is used.

	Findings	**Imi_grz_data**
Number of datasets	171.932	187.651
Primary key table 1	*FirstNo, SupplementNo*	
Number of joined datasets	171.932	
Matched datasets	6.640	
Matched datasets after revision	171.932	
Final result of matched datasets	171.932	
100 % of the datasets is contained in one of the compared tables	✓	

Table 3.22:Results of the comparison between the Imipatho and MSA DB: The MSA table findings is compared with the Oracle 10g table imi_grz_data. The table shows that all of the datasets contained in findings can be joined with imi_grz_data. The primary key including "Befundnummer" and "Nb"is the same for both of the tables and used for the join. After the simple comparison about 4 percent of the data is equal. A revision of the comparison shows that the data of the findings table is fully contained in the imi_grz_data table.

Table 3.22 shows that table *findings* is fully contained in table *imi_grz_data*. The primary key of both of the tables consists of the fields *befundnummer* and *nb*. After joining and comparing them by the use of the primary key only about four percent of the datasets are identical. The revised comparison, see table 3.23 reveals that all the information is contained in the Oracle 10g DB table.

Identifier	**Replaced characters**	**Replaced by**
Underlying disease, coroner's inquest, additional desease, Histo, micro, diagnose, material	All characters except alphanumeric and numeric characters. Additionally, alphanumeric characters are converted to uppercase. Umlauts and characters with accent signs get replaced with the corresponding Latin character. The sharp S is replaced by an empty string.	Empty string, Ä, À, À→ A Ö, Ò, Ò→ O Ü, Ù, Ú → UE Í → I È, É → E
Tnmt	All characters except alphanumeric and numeric characters.	Empty string

Table 3.23: Adaptions for the revised comparison: The two compared tables "findings" and "imi_grz_data" contain different character sets. To be able to compare the content of the two tables characters have to be replaced in the fields.

After the comparison, three tables remain for the migration of the MSA and Oracle data sources:

- *Imi_grz_data*,
- *Imi_aura_data*
- *Tissues*

The structure of the three tables is similar but the primary keys contain data in different formats. Thus, new primary keys have to be introduced to allow a comparison between the tables, see figure 3.8 and table 3.24.

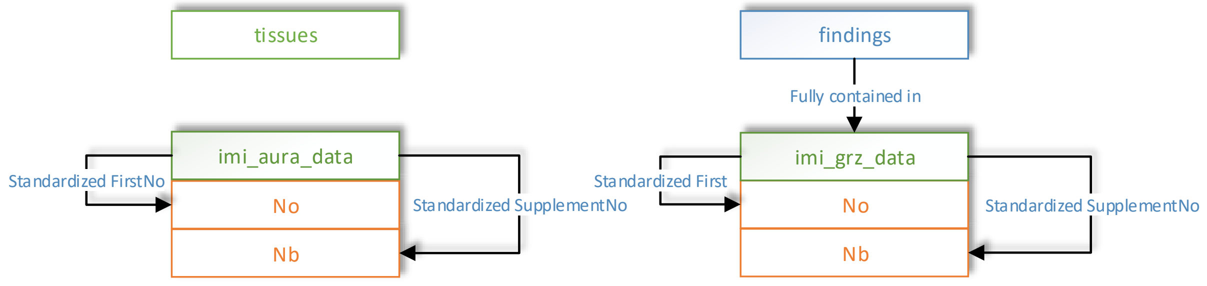


Figure 3.8: Intermediary Result of the cleansing in this phase: Table "findings" is contained in table "imi_grz_data". "imi_aura_data" and "imi_grz_data" are extended by the standardized columns "No" and "Nb" for further comparison. Table "tissues" is already extended by the mentioned columns, see section 3.5.2.3.

	Imi_grz_data	**Imi_aura_data**	**Tissues**
Primary key fields	*FirstNo, SupplementNo*		*No, SupplementNo*
Format of field 1	10 digits	8 digits	
Example of field 1	2016008061	16008061	
Format of field 2	1 digit		
Example of field 2	2		
New common key	*No*: 16008061, *Nb*: 2		
Transformation rule for field 1	Last eight digits of *FirstNo*	All digits of *FirstNo* respictievly *No*	
Transformation rule for field 2	*SupplementNo* contains the common format		

Table 3.24: Overview of the transformation rules for the new primary keys: The primary keys of the three remaining source tables exist in different formats. To be able to compare the tables it is necessary to create a common primary key for all the tables. The new primary key consists of the two columns No and Nb. In "Imi_grz_data" the "FirstNo" has to be adapted and copied into "No" by snipping the first two digits of the field. "FirstNo" of "Imi_aura_data" is copied without modification and the table "Tissues" contains the field, see section 3.5.2.3. The supplement no can be used without being modified is and copied into the new field "Nb".

Table 3.25 illustrates the contained data of the three remaining source tables.

Alias	Imi_grz_data	Imi_aura_data	Tissues
Information about the HCI			
Surname	✗	✗	✓
Forename	✗	✗	✓
DateOfBirth	✗	✗	✓
PatientAge	✓	✓	✓ Calculable
PatientSex	✓	✓	✗
ExaminationDate	✗	✗	✓
ExaminationYear	✓	✓	✓
Department	✓	✗	✗
No	✓	✓	✓
LastNo	✓	✓	✓
Organ	✓	✓	✓
Sender	✓	✓	✓
Information about the HNO			
Nb	✓	✓	✓
ExaminationType	✓	✓	✓
Diagnose	✓	✓	✓
HistologicResult	✓	✓	✗
MacroscopicResult	✓	✓	✗
UnderlyingDisease	✓	✓	✗
AdditionalDisease	✓	✓	✗
CoronersInquest	✓	✓	✗
IcdO	✓	✗	✗
Icd10	✓	✗	✗
Parameters			
Tumor staging parameters			

TNMM	✓	✗	✗
TNMN	✓	✗	✗
TNMR	✓	✗	✗
TNML	✓	✗	✗
TNMV	✓	✗	✗
TNMP	✓	✗	✗
TNMT	✓	✗	✗
HER2 parameters			
HER2 / CEP 17	✗	✗	✓
Ratio	✗	✗	✓
Her2 /Dako	✗	✗	✓
Ratio complete	✗	✗	✓

Table 3.25: Overview of the stored information in the source tables: The table divides the information about the datasets by the contents of the Histocircle and the finding with its parameters. Since all the three tables contain different information the joined datasets are used for comparison and merging information.

With the newly generated common primary key the three tables are joined together and the corresponding fields are compared. Additionally, each of the tables is separately joined and compared with the other ones. The joined datasets are marked in their sources by introducing a new column named *merge* and inserting an "X" in it. Figure 3.9 illustrates the joining of the tables.

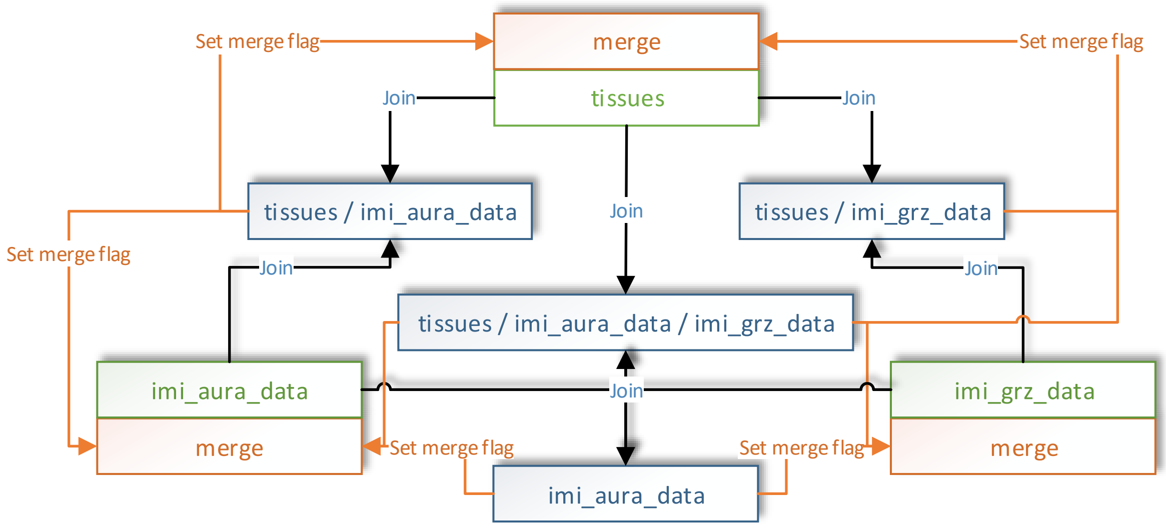


Figure 3.9: Overview of the joining of the source tables for data cleansing and merging: The three source tables are joined together and form the resulting table "tissues / imi_aura_data / imi_grz_data". Additionally, each table is joined with the other ones separately. In this way four additional tables (the blue ones) are created. The datasets contained in the newly generated tables are marked in the source tables by setting an "X" into the merge flag column.

By applying this procedure, four additional tables are created. The data contained in the seven source tables is cleansed by the following steps:

1. The newly generated tables are extended by the migration flag column already contained in the source tables.
2. The newly generated tables are checked for consistency of the information about the HCI by adding a column for each information contained twice or more across the joined tables.
3. Inconsistencies related to the HCI are marked by setting an "X" into the corresponding flag column.
4. Across all tables, all datasets are updated by setting the "X" into the migration flag column where one or more inconsistency flags of the HCI are set.
5. The seven source tables are extended by new flag columns to be able to mark inconsistencies related to the HNO. Flag columns are created for all the information, which occur in two or more tables like the diagnosis, examination type, histologic result, etc.
6. Inconsistences related to the finding are marked with an "X" in the corresponding datasets of the checked source.
7. The migration flag of all the datasets containing a mark in one or more of the flag columns related to the HCI and finding is set to an "X".

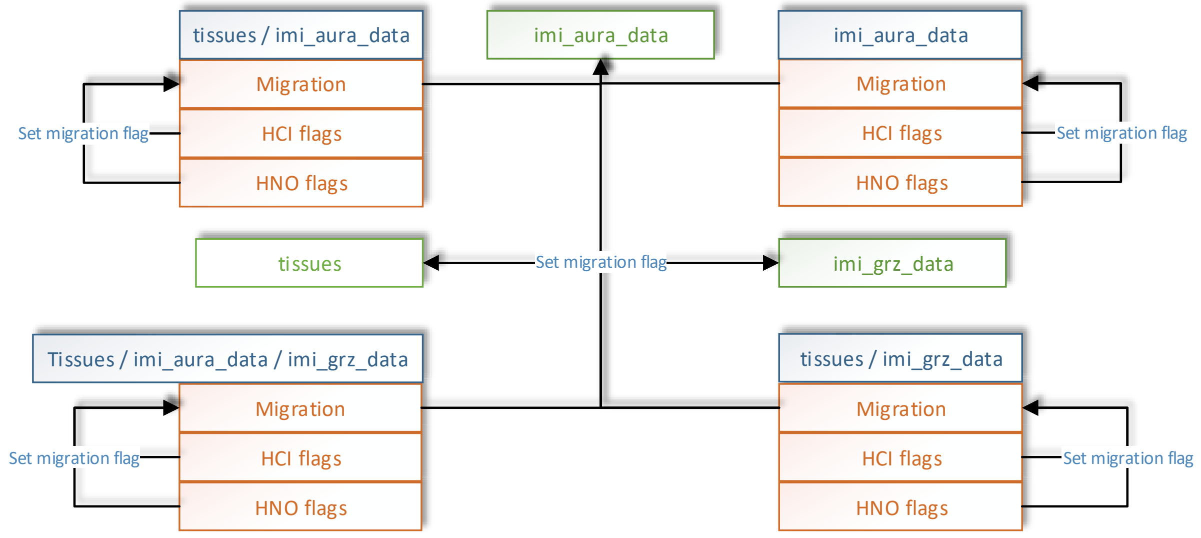


Figure 3.10: Overview of the cleansing steps in this phase: The newly generated join tables are extended by the flag column "migration" and flag columns for the HCI and the HNO. The corresponding flag of each table is set if the HCI or the HNO comparison reveals inconsistencies. The migration flag of each of the datasets is set if one of the HNO or the HCI flags was set. The migration flag of the original three tables marked in green is set if a dataset with issues of the join tables is contained.

Figure 3.10 illustrates the overview of the cleansing steps. After these steps all the datasets containing issues are marked. The rest of the datasets is ready for the migration. To be able to compare equivalent data with different charsets, the comparison of the datasets is adapted. Some of the equivalent data cannot be compared because of different formats of the data sources, see table 3.26.

Alias / ID	**Imi_grz_data**	**Imi_aura_data**	**Tissues**
Information about the HCI			
LastNo	✓	✓	✓
PatientAge	✓	✓	✓ Age +/- 1
PatientSex	✓	✓	✗
Organ	✗	✓ Ä → AE Ö → OE Ü → UE Only alphanumeric values	✓ Ä, [→ AE Ö, \ → OE Ü,] → UE ß, ~ → SS Only alphanumeric values
ExaminationYear	✓	✓	✗

Sender	✓	✓	✗
Information about the HNO			
Diagnose	✓	✓	✓
DiagnoseMolecular	✓	✓	✗
HistologicResult	✓	✓	✗
UnderlyingDisease	✓	✓	✗
AdditionalDisease	✓	✓	✗
CoronersInquest	✓	✓	✗
ExaminationType	✓	✓	✓ P0 → PO, P~, P$ → NULL

Table 3.26: Overview of the compared columns and adaptions for the comparison: To be able to compare the equivalent data adaptions have to be made. The fields marked in green are compared to each other in the corresponding category respectively ID. Fields marked with the cross are not suitable for comparison or not available in the table.

3.5.2.6 *Cleansing within the MySQL database*

This DB contains one table containing information about FFPE samples from 2009 to 2015. Similar to the table *tissues*, see section 3.5.2.3, a new temporary table with the distinct information about the HCIs of the MySQL table is generated. The original table is extended by the column “migration”. All the datasets with the *FirstNos* contained multiple times in the temporary table are marked by setting an “X” into the migration flag column, see figure 3.11.

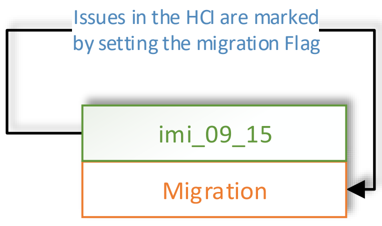


Figure 3.11: Result of the cleansing within the MySQL database: The DB contains one table. The column “Migration” is added to the table. The HCIs contained in the tables are compared and tested for inconsistencies. Datasets containing issues are marked with an “X” in the migration flag column.

3.5.2.7 *Cleansing of the XML sources*

Each XML file contains the current information about the HCIs when the XML file was extracted by the IMI. The latest XML file contains the newest information about the HCI and older information contained in the DB has to be updated. Thus, the update respectively the cleansing of the XML files takes place during the import of the files into the target Oracle 11g DB. The XML files are imported chronologically in ascending order. The migration algorithm checks if the current dataset of the read out XML source is already contained in the target DB. If this is the case the information is updated to the state of the dataset of the read out XML file.

3.5.3 Design phase

The design phase covers two important tasks:

1. Specification of the mapping between the source and the target DB, see section 3.5.3.1
2. Implementation of the migration algorithm, based on the mapping specifications, see section 3.5.3.2

3.5.3.1 Mapping specification

Figure 3.12 - figure 3.15 illustrate the mapping specifications of the design phase in detail. The target DB distinguishes between information about the HCI, HNO and lookup and linking information. This kind of data is stored in separate tables of the target Oracle 11g DB.

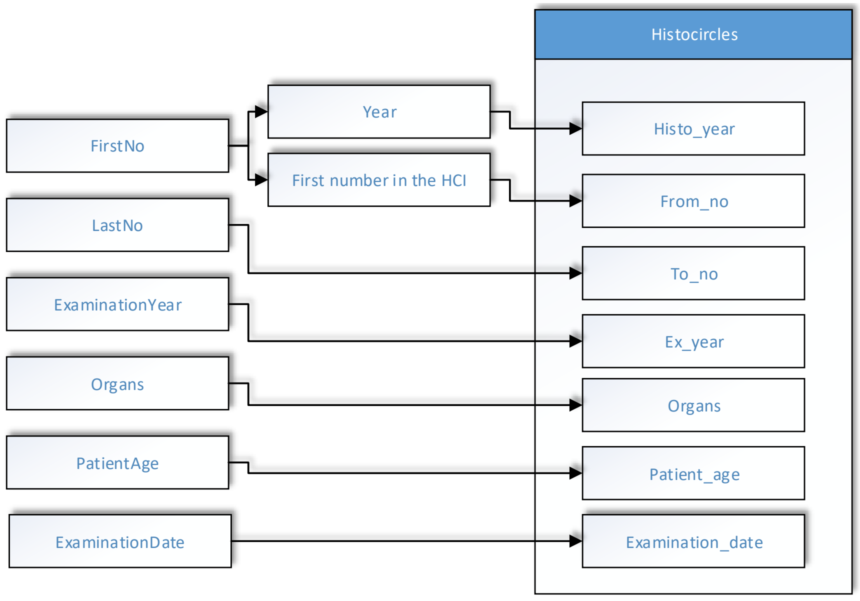


Figure 3.12: Mapping information about the HCI: The information about the HCI is stored in the table "Histocircles" of the target database. The source column "FirstNo" contains information about the year and the number of the HCI and is stored separately in the target Oracle 11g database.

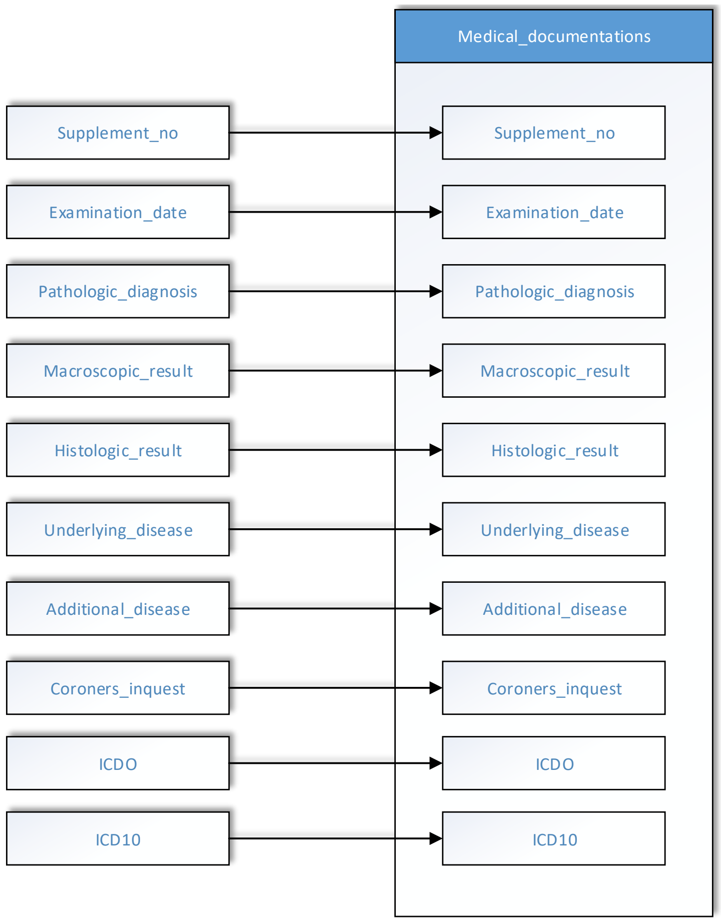


Figure 3.13: Mapping information about the HNO: The information about the HNO is stored in the table "Medical_documentations" of the source Oracle 11g database.

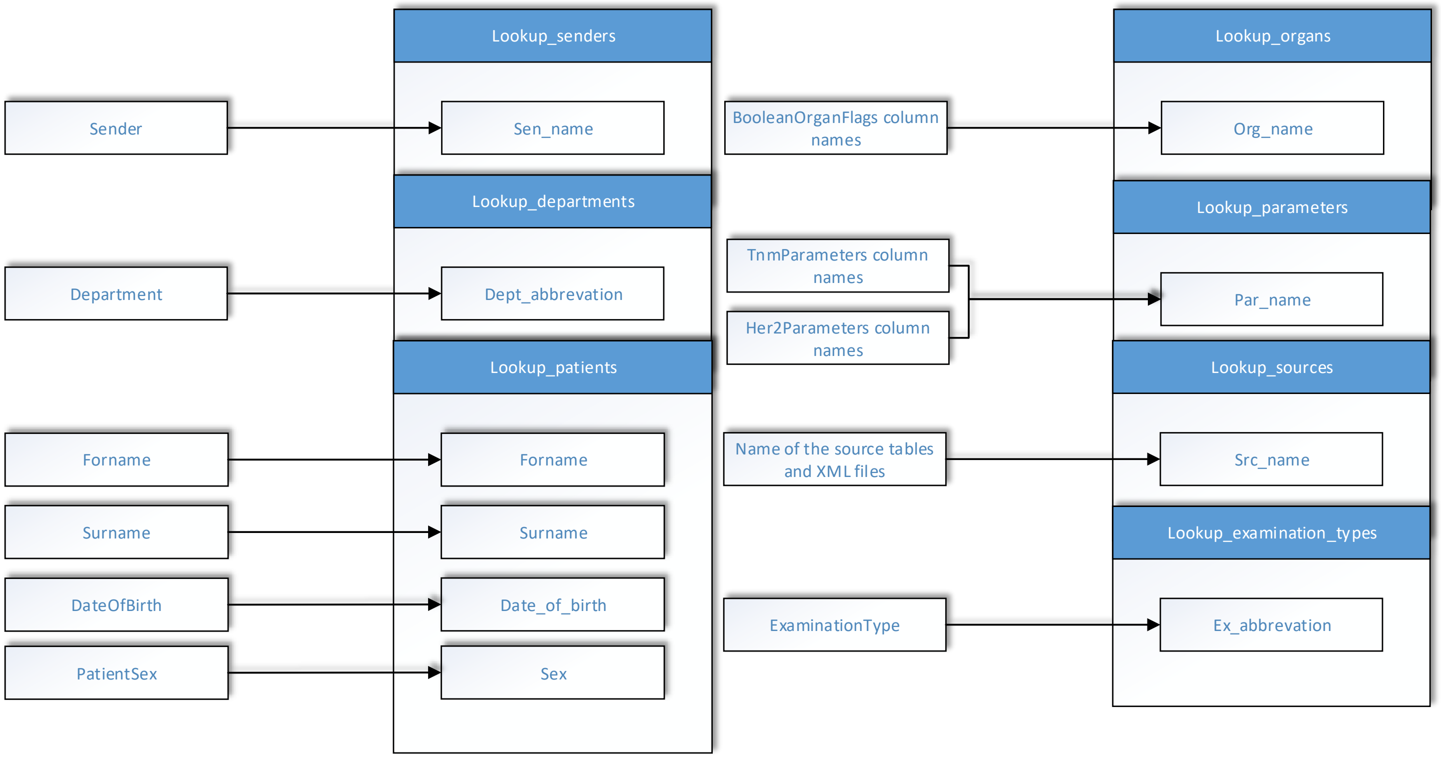


Figure 3.14: Mapping of the lookup information: The information about the sender, department, patient, organ, source name, examination type is stored in the corresponding lookup tables of the source database. Additionally, the available parameter types and the categorized organs are stored in further lookup tables. The primary keys of the lookup up tables are linked to the foreign keys in other tables of the target DB, see figure 3.2.

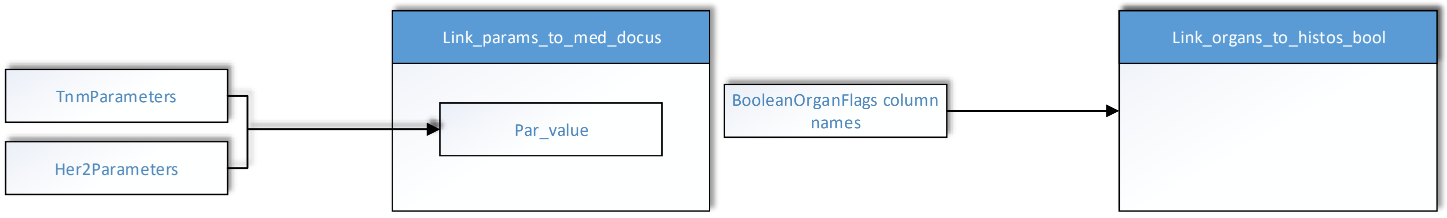
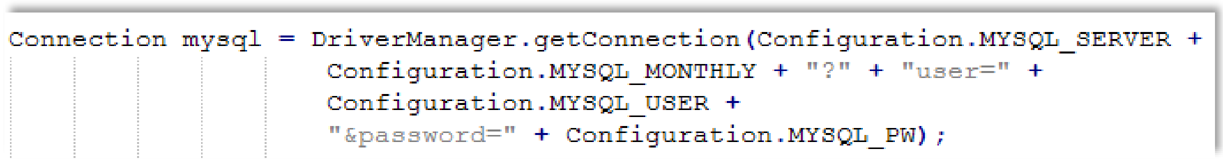


Figure 3.15: Mapping of the lookup information: The values of the parameters of the tumor staging and HER2 are stored in a so called link table of the database. Additionally the information about the "BooleanOrganFlags" are stored in the target table "Link_organs_to_histos_bool". This table combines the foreign keys of the HCI and the categorized organ to maintain the information about the sources.

3.5.3.2 Implementation of the migration

The migration itself is implemented in Java by use of the Eclipse IDE. Each source table and file is migrated separately from the source into the target DB. The XML sources are cleansed during migration and the sources of the DBs are cleansed in the previous phase, see section 3.5.2. In general, the implemented algorithm for migration consists of the following steps:

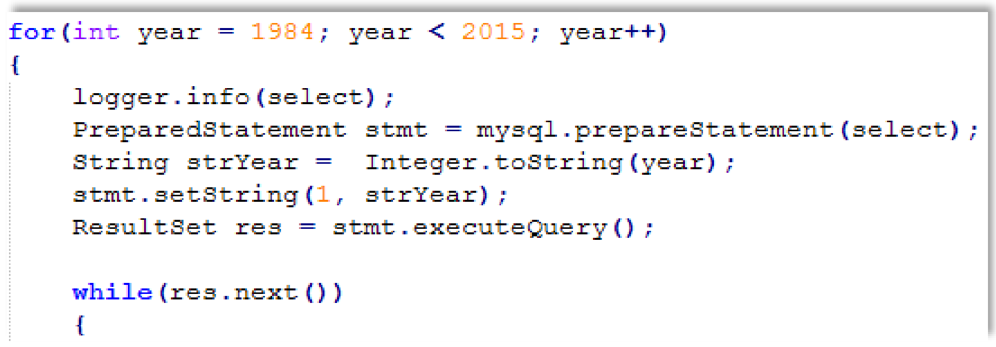
1. A connection is established to the source - if the source is a DB table - by the use of the imported JDBC driver, see listing 3.3



```
Connection mysql = DriverManager.getConnection(Configuration.MYSQL_SERVER +
                    Configuration.MYSQL_MONTHLY + "?" + "user=" +
                    Configuration.MYSQL_USER +
                    "&password=" + Configuration.MYSQL_PW);
```

Listing 3.3: Establishment of the connection to one of the source DBs

2. The data is extracted from the source. In the case of a source DB table the data is extracted by the use of the divide and conquer paradigm: Each examination year of the sources is loaded separately into memory for further processing, see listing 3.4

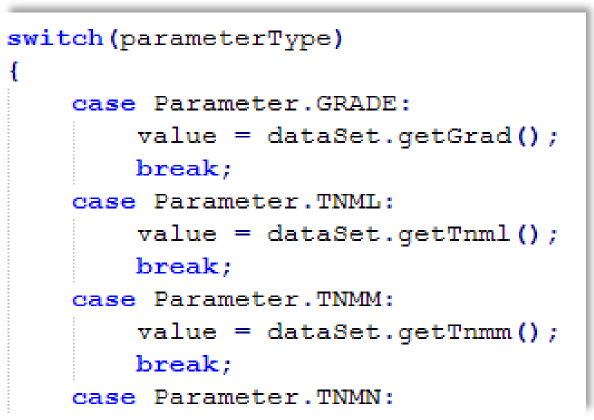


```
for(int year = 1984; year < 2015; year++)
{
    logger.info(select);
    PreparedStatement stmt = mysql.prepareStatement(select);
    String strYear =  Integer.toString(year);
    stmt.setString(1, strYear);
    ResultSet res = stmt.executeQuery();

    while(res.next())
    {
```

Listing 3.4: Extraction of the source data: The datasets are fetched separately for each year in the source table. The resulting datasets are further processed one after another in the while loop.

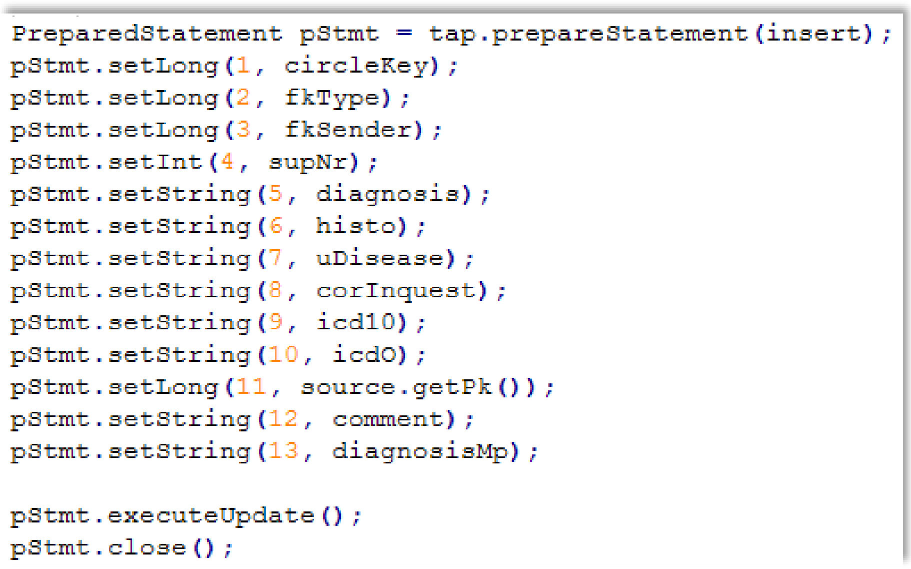
3. After the extraction each dataset is handled separately for further processing.
    a. A connection is established to the target Oracle 11g DB.
    b. It is checked if the HCI already exists.
4. The extracted dataset has to be transformed as specified in the mapping specifications to fit into the new schema of the target DB, see section 3.5.3.1. First the information about the HCI is transformed if the HCI is not contained in the target DB yet. Afterwards the information about the HNO is transformed. Listing 3.5 shows an example of the parameter mapping of an HNO. XML sources are additionally checked for inconsistencies related to the information about the HCI. Datasets with issues regarding the HCI are not migrated and logged in the migration log file. Furthermore, XML sources can contain updated information about datasets already stored in the DB. Thus, existing datasets regarding about the HNO are updated in the target DB.



```
switch(parameterType)
{
    case Parameter.GRADE:
        value = dataSet.getGrad();
        break;
    case Parameter.TNML:
        value = dataSet.getTnml();
        break;
    case Parameter.TNMM:
        value = dataSet.getTnmm();
        break;
    case Parameter.TNMN:
```

Listing 3.5: Mapping of the parameters of the HNO: Depending on the parameter type of the source, the corresponding value of the source is fetched to be later inserted correctly in the "link_params_to_med_docus" table of the target DB.

5. After completion of the transformation, the dataset is loaded into the target DB by the use of the connection already opened in step 3. Listing 3.6 illustrates how values are supplied, the execution and the closing of the prepared insert SQL statement.



```
PreparedStatement pStmt = tap.prepareStatement(insert);
pStmt.setLong(1, circleKey);
pStmt.setLong(2, fkType);
pStmt.setLong(3, fkSender);
pStmt.setInt(4, supNr);
pStmt.setString(5, diagnosis);
pStmt.setString(6, histo);
pStmt.setString(7, uDisease);
pStmt.setString(8, corInquest);
pStmt.setString(9, icd10);
pStmt.setString(10, icdO);
pStmt.setLong(11, source.getPk());
pStmt.setString(12, comment);
pStmt.setString(13, diagnosisMp);

pStmt.executeUpdate();
pStmt.close();
```

Listing 3.6: Execution of the prepared insert statement: The prepared statement string stored in the "insert" variable is supplied with the parameter's source HNO data. Afterwards the query is executed and closed.

6. If the final dataset of a source is inserted a commit of the session is executed.
7. Both the connections to the source and target DB are closed.

3.5.4 Migration phase

The migration phase covers the execution of the implemented program, see section 3.5.3.2. In general, the software processes the Extract, Transform and Load (ETL), which means that the data is extracted from the sources, transformed to fit into the new schema of the target and loaded into the Oracle 11g target DB. The software creates a log file containing information about datasets, which could not be imported into the target. If errors emerge during the migration of the cleansed DB sources the implementation of the migration algorithm is revised because all possible issues were eliminated in the profile and audit phase, see section 3.5.2.

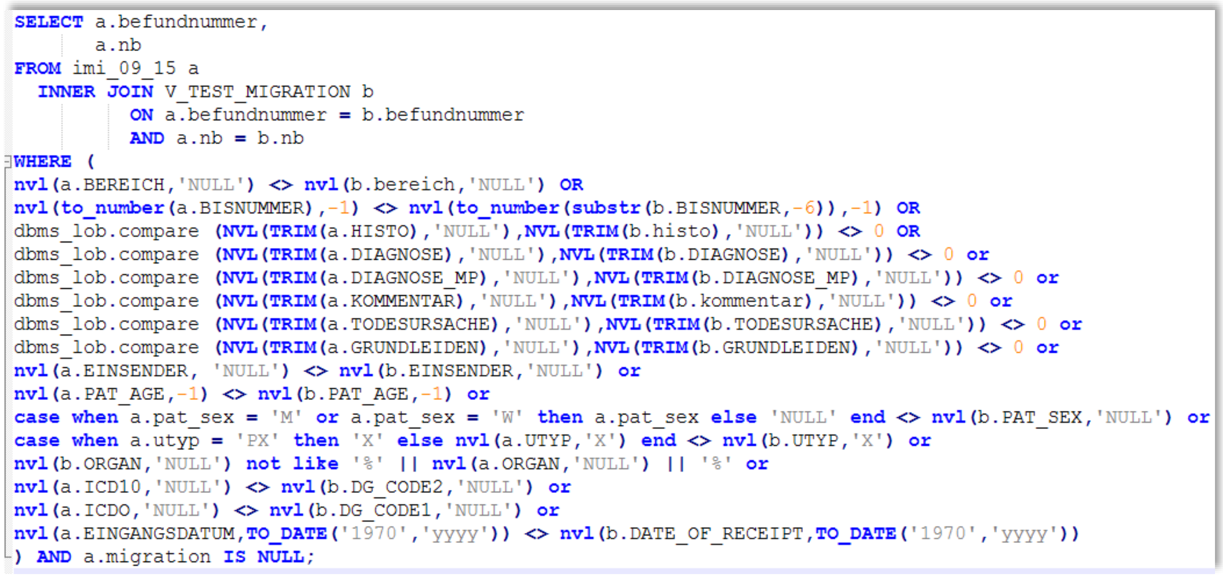
3.5.5 Testing phase

After the migration in the previous phase the migrated data has to be tested for correctness and completeness. All the valid migrated sources have to be tested to ensure that the migration process was successful.

3.5.5.1 Testing the SQL sources

In the case of the practical example, the test phase is executed in the target Oracle 11g database by the use of SQL. The following steps are executed to confirm that the sources have been successfully migrated:

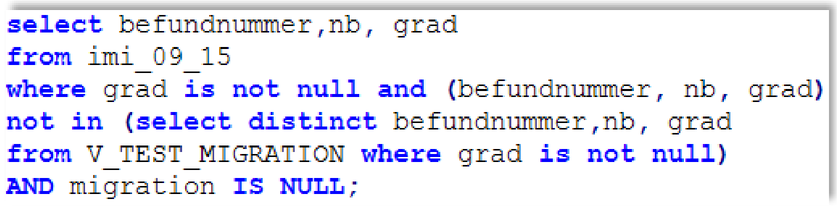
1. All the four remaining and extended SQL sources of the profile and audit phase, see section 3.5.2, are imported from the source into the target DB by the use of the Data Pumper of SQL Workbench / J.
2. In the target Oracle 11g DB a DB view is created whose structure is similar to the imported SQL sources by the use of inner and outer joins.
3. For each of the four sources test scripts for the HCI and HNO with their *TnmParameters* and *HER2Parameters* are created in a way that lists wrong datasets with their *FirstNo* and *SuplementNo*. The testing scripts are extended by the migration rules defined during the design of the migration to approve valid testing. Additionally, the number of migrated datasets is checked to ensure that all the valid datasets of the profile and audit phase are migrated.
4. If the test scripts are processed without showing wrongly migrated datasets the migration for the SQL sources is successfully completed. In case of wrongly migrated datasets the migration algorithm has to be revised. Analyses have to be made to figure out where the error happened and a step back to the design phase has to be made.



```
SELECT a.befundnummer,
       a.nb
FROM imi_09_15 a
  INNER JOIN V_TEST_MIGRATION b
          ON a.befundnummer = b.befundnummer
          AND a.nb = b.nb
WHERE (
nvl(a.BEREICH,'NULL') <> nvl(b.bereich,'NULL') OR
nvl(to_number(a.BISNUMMER),-1) <> nvl(to_number(substr(b.BISNUMMER,-6)),-1) OR
dbms_lob.compare (NVL(TRIM(a.HISTO),'NULL'),NVL(TRIM(b.histo),'NULL')) <> 0 OR
dbms_lob.compare (NVL(TRIM(a.DIAGNOSE),'NULL'),NVL(TRIM(b.DIAGNOSE),'NULL')) <> 0 or
dbms_lob.compare (NVL(TRIM(a.DIAGNOSE_MP),'NULL'),NVL(TRIM(b.DIAGNOSE_MP),'NULL')) <> 0 or
dbms_lob.compare (NVL(TRIM(a.KOMMENTAR),'NULL'),NVL(TRIM(b.kommentar),'NULL')) <> 0 or
dbms_lob.compare (NVL(TRIM(a.TODESURSACHE),'NULL'),NVL(TRIM(b.TODESURSACHE),'NULL')) <> 0 or
dbms_lob.compare (NVL(TRIM(a.GRUNDLEIDEN),'NULL'),NVL(TRIM(b.GRUNDLEIDEN),'NULL')) <> 0 or
nvl(a.EINSENDER, 'NULL') <> nvl(b.EINSENDER,'NULL') or
nvl(a.PAT_AGE,-1) <> nvl(b.PAT_AGE,-1) or
case when a.pat_sex = 'M' or a.pat_sex = 'W' then a.pat_sex else 'NULL' end <> nvl(b.PAT_SEX,'NULL') or
case when a.utyp = 'PX' then 'X' else nvl(a.UTYP,'X') end <> nvl(b.UTYP,'X') or
nvl(b.ORGAN,'NULL') not like '%' || nvl(a.ORGAN,'NULL') || '%' or
nvl(a.ICD10,'NULL') <> nvl(b.DG_CODE2,'NULL') or
nvl(a.ICDO,'NULL') <> nvl(b.DG_CODE1,'NULL') or
nvl(a.EINGANGSDATUM,TO_DATE('1970','yyyy')) <> nvl(b.DATE_OF_RECEIPT,TO_DATE('1970','yyyy'))
) AND a.migration IS NULL;
```

Listing 3.7: Test script of the MySQL source: The datasets of the MySQL DB are tested against the migrated data of the Oracle 11g DB. Only valid datasets are tested. This is achieved by joining only the datasets where the migration flag of the source is not set. After successful execution of the script no results are returned from the SQL query. In case of a result the source is not correctly migrated.

Listing 3.7 illustrates the testing script of the MySQL source without testing of the parameters. The source table is tested against the migrated data. Listing 3.8 illustrates the testing script of one tumor staging parameter, which is similar to the other parameters.

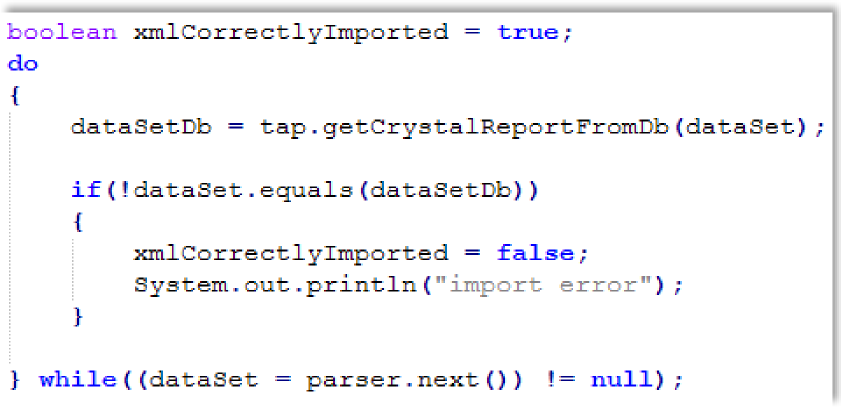


```
select befundnummer,nb, grad
from imi_09_15
where grad is not null and (befundnummer, nb, grad)
not in (select distinct befundnummer,nb, grad
from V_TEST_MIGRATION where grad is not null)
AND migration IS NULL;
```

Listing 3.8: Test script of the parameters: Each migrated parameter is tested against the valid datasets in the source. If a result with datasets is returned the migration algorithm is not implemented correctly and has to be revised.

3.5.5.2 Testing the XML sources

The XML sources are tested during the processing of the migration. Each XML file is imported into the Oracle 11g target DB and tested afterwards. Listing 3.9 shows a code snippet of the XML source test.



```
boolean xmlCorrectlyImported = true;
do
{
    dataSetDb = tap.getCrystalReportFromDb(dataSet);

    if(!dataSet.equals(dataSetDb))
    {
        xmlCorrectlyImported = false;
        System.out.println("import error");
    }

} while((dataSet = parser.next()) != null);
```

Listing 3.9: Test of the imported datasets of the XML sources: One dataset after the other is tested within the while loop. The dataset stored in the target DB is fetched and compared with the dataset of the XML source. If they are not equal, the XML source was not correctly imported and the corresponding flag is set to false.

Basically, the XML sources are tested by applying the following steps:

1. After completion of the import of an XML source, one dataset after another is stored into a Java object.
2. With the object a DB query is executed, fetching the stored dataset of the target DB by asking for the corresponding *FirstNo* and *SupplementNo*.
3. The fetched result is converted into the same object as the source.
4. The source and the target object are compared with each other. If differences between the two objects are recognized the design respectively the implementation have to be revised.
5. If all the target objects are equal to the source the migration of the XML file was successful.

3.5.6 Finalization

After the successful migration the sources are archived by creating SQL dumps in case of the Oracle and MySQL sources. The MSA DB file, the XML sources and the documentation of the migration project are copied into the archive.

3.5.7 Overview

At the beginning of the migration process. a MSA DB, two Oracle DBs, a MySQL DB and five XML files are important for the migration. The different sources are checked for redundancies during the data cleansing.

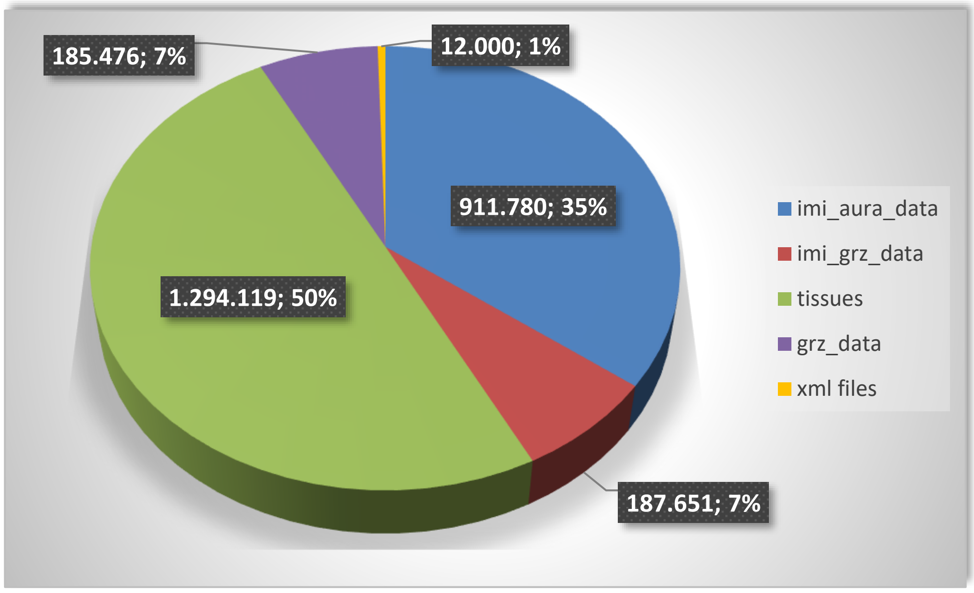


Figure 3.16: Overview of the remaining sources for migration: The MSA DB containing the "tissues" table with about 1.3 millions of datasets represents 50 percent of the source datasets. The second Oracle DB contains two important tables for migration with about 1.1 million of datasets. The Oracle DB 2 makes up 42 percent. Table "grz_data" of the MySQL source contains about 190 thousand datasets representing about 7 percent of the remaining migration sources. 12.000 datasets are contained within the five XML source files, representing 1 percent of the source datasets.

After the cleansing the following sources remain for the migration process:

- Table *tissues* from the MSA source
- Table *imi_aura_data* from the Oracle DB 2 source
- Table *imi_grz_data* from the Oracle DB 2 source
- Table *grz_data* from the MySQL DB source
- Five XML files

The information stored in the Oracle DB 1 source is completely contained within the MSA sources. Figure 3.16 illustrates the sources for migration and the number of datasets they contain. It reveals that half of the source data is contained in the MSA DB, 42 percent is contained in the Oracle DB 2, 7 percent is contained within the MySQL source and 1 percent of the information is provided by the XML sources. The mentioned sources partially contain inconsistent data. After the detection all the datasets containing inconsistencies are marked by the introduction of new columns. Figure 3.17 illustrates that only a small amount of the datasets contains issues within the MSA and Oracle sources. 100 percent of the MySQL and XML sources are migrated into the source DB.

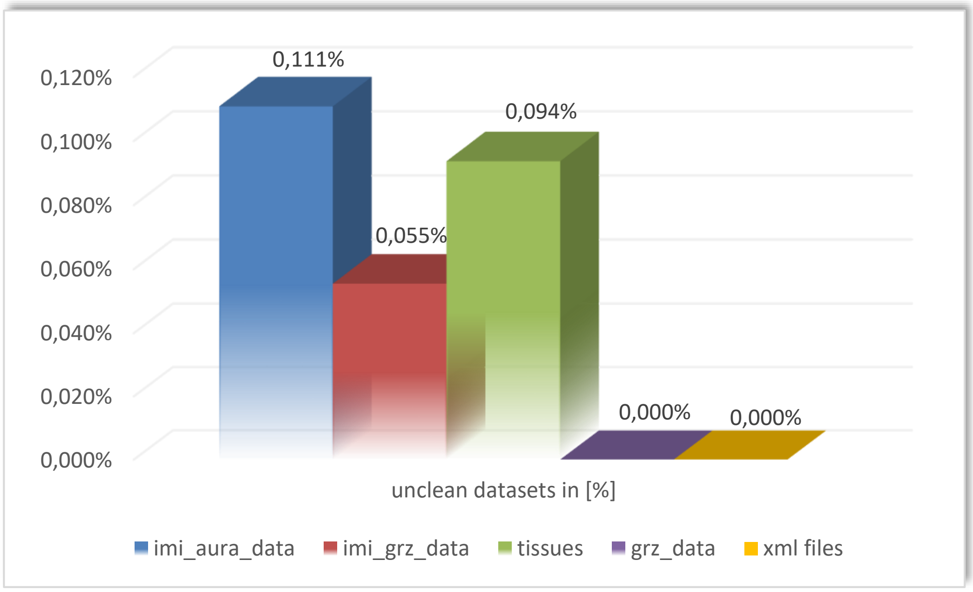


Figure 3.17: Overview of unclean datasets within the source tables in percent: 100 percent of the XML and MySQL sources are consistent and are migrated into the target Oracle 11g DB. About 0,1 percent of the datasets in the tables "imi_aura_data" and "tissues" are inconsistent. About 0,055 percent of the datasets within the "imi_grz_data" table are not migrated because of issues.

Figure 3.18 shows the total amount of the datasets which are included in the sources and the number of datasets which are migrated into the newly designed Oracle 11g DB. The sum of the migrated datasets is not the equivalent to the actual number of datasets in the target DB, because datasets in the different sources are partially merged together. Additionally, an HCI can be contained in more than one XML source file meaning that the newer dataset is the valid one overwriting the previous one.

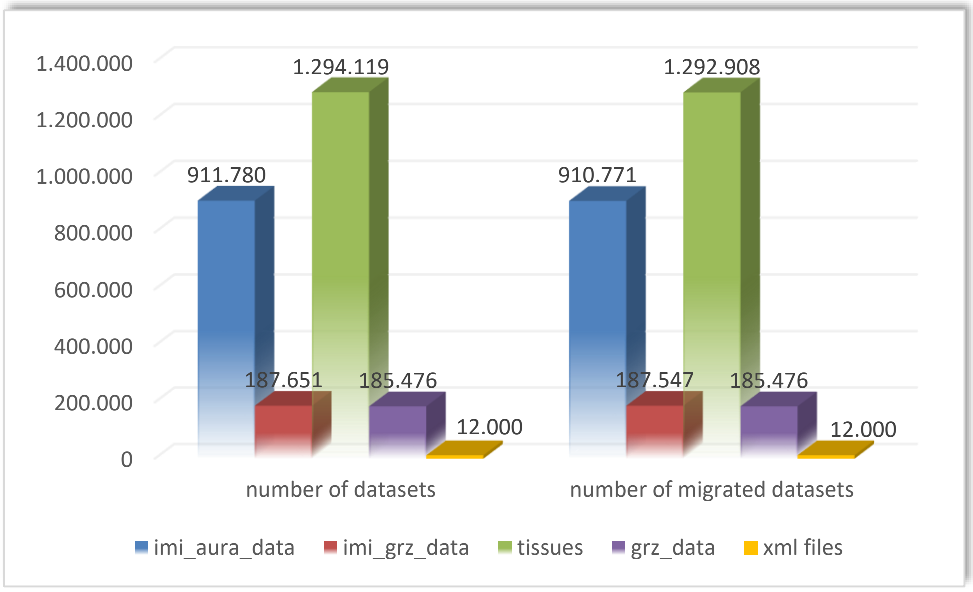


Figure 3.18: Overview of the number of datasets contained and migrated from the sources

After cleansing, the algorithm containing the mapping from the sources to the target is specified and implemented in Java. The mapping shows that redundant data of the HCI containing more than one HNO is eliminated by introducing separate tables for the HCI and the HNO: HNOs, which belong to the same HCI are linked by the use of the corresponding foreign key of the HCI. After the execution of the migration algorithm 1.829.843 HNOs are listed within the table *medical_documentations* of the target Oracle 11g DB. The result of the tests is that all the datasets marked for migration are successfully migrated into the target environment.

4 Discussion

In the process of creating this thesis a migration of different DBs of different DBMS and XML files containing samples of tissue was implemented and executed. The different sources and the target DB are used by the Biobank of Graz, an institute of the Medical university of Graz.

The first step was the identification of the data stored in the different sources. Additionally, the relevant data for migration was selected. Equipped with the knowledge about which data has to be stored in the future the target DB was designed and implemented in Oracle 11g. Afterwards, the migration process was started. Initially the migration strategy and the platform for migration were selected. The Big-bang approach - used in this thesis - offers short migration times and no need for synchronization. After the initialization the next step was the cleansing phase where inconsistencies between overlapping data of the sources was identified and eventually cleared. The sources contained about 2.5 million datasets where about 2 thousand datasets were detected as inconsistent. The inconsistent data was not migrated and marked for a later manual revision. Additionally, redundant data was identified and marked to be not migrated in this phase of the migration process. The cleansing phase was executed by the use of SQL. Afterwards the migration algorithm was designed and implemented. In the design phase the data mapping was specified. The algorithm was implemented in Java which can run on every platform where the JVM is installed. During the migration the data is extracted from the sources, transformed to fit into the new DB and loaded into the target relational Oracle 11g DB. After the successful one-time migration process the migrated data was successfully tested for correctness and completeness by comparing it with the datasets of the sources in SQL.

4.1 Comparison of the target DB with the sources

The source data was spread between five different types of sources. The current version of the DB contains all the data in one source, which allows the user to search faster for suitable data. Lots of the spread information about the sources could be merged in the target DB in a way that brought the different pieces of information together to provide the full information by one query. The structure of the new Oracle 11g DB is stricter than the previous ones by the use of newly introduced constraints. The current structure distinguishes between information about the HCI and HNO which are related to each other. Thus, the amount of data stored in the DB is reduced by offering the complete information about the source datasets. Furthermore, it is more difficult to insert data causing redundancies and inconsistencies.

According to additional comparisons an improvement of the execution speed of queries should be achieved. Average execution times of MSA queries are not performed as fast as queries in Oracle. Conversely it should be acknowledged that the average CPU utilization and memory usage is higher in Oracle. This negative aspect probably will not be of great consequence, seeing that the target Oracle 11g DB runs on a server which usually provides better performance and more memory than client computers (59,60).

4.2 Conclusion

This thesis pointed out that data is an important resource for companies. It was shown that the management of data is a critical objective to remain competitive. To be able to manage data efficiently it is necessary to migrate outdated systems to adequate state of the art systems. It was pointed out that migrations can be processed by applying several approaches depending on the complexity. Finally, the most important factors of data migration projects were discussed. The objective of implementing a migration for several data sources complying with the requirements, see section 1.12 was achieved. A new database was designed and implemented in Oracle 11g offering the following enhancements:

- All the data is accessible through one data source.
- Better structure by distinguishing between information about the clinical cases, findings, clinical senders, clinical departments, etc., see section 3.4.
- Redundancies and duplicated datasets are significantly reduced by the introduction of lookup tables.
- Improvement of the stored datatype have been achieved - e.g. a date is now stored as date instead of a string.

The new version of the database is an in-house product offering benefits like the full control of the sources, independency and the possibility to extend the DB by new features. Through the migration process the data contained in the target DB was cleansed. The transformation improved the data classification by transforming strings containing only numeric values into numbers. Later XML files containing the same structure of the XML sources can be easily imported into the new Oracle DB by applying the implemented migration software.

4.3 Outlook

An interesting point for improvement of the current version of the migration software would be an easier configuration of the process by offering e.g. a properties file where the sources can be inserted by declaring the connection and the data structure. Additionally, the mapping specifications could be inserted in the properties file. After the handover of the properties file to the software the migration process could be executed. At this point in time the sources and the mapping have to be altered within the Java source code. An improvement regarding the new Oracle 11g DB would be the deletion of the organs text field. Instead of this field, linking the clinical cases to the organs lookup table would reduce storage and processing time when organs are searched. To achieve such an improvement, the content of the organs field has to be cleansed by e.g. mapping misspelled organs to the corresponding one in the organs lookup table.

5 Bibliography

1. Singh SK. Database systems concepts, design, and applications. New Delhi, India: Dorling Kindersley (India); 2006.

2. Connolly TM, Begg CE. Database systems: a practical approach to design, implementation, and management. 4. ed. Harlow: Addison-Wesley; 2005. 1374 p. (International computer science series).

3. Kedar,Seema. Database Management System. Pune: Technical Publications Pune; 2009.

4. Paterson J, Edlich S, editors. Comparing the Object and Relational Data Models. In: The definitive guide to db4o: [the first comprehensive guide to db4o, the open source native object database for NET and Java]. Berkeley, Calif: Apress; 2006. p. 31–46. (The expert's voice in open source).

5. Navathe SB. Evolution of data modeling for databases. Commun ACM. 1992;35(9):112–123.

6. DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems [Internet]. [cited 2015 Dec 30]. Available from: http://db-engines.com/en/

7. DB-Engines Ranking - popularity ranking of database management systems [Internet]. [cited 2015 Dec 30]. Available from: http://db-engines.com/en/ranking

8. historical trend of the popularity ranking of database management systems [Internet]. [cited 2015 Dec 30]. Available from: http://db-engines.com/en/ranking_trend

9. Janssen C. Data Migration [Internet]. Data Migration. [cited 2015 Feb 22]. Available from: http://www.techopedia.com/definition/1180/data-migration

10. Morris J. Practical Data Migration [Internet]. Swindon: BCS Learning & Development Ltd.; 2012 [cited 2015 Feb 22]. Available from: http://public.eblib.com/choice/publicfullrecord.aspx?p=1010226

11. Types of data migrations [Internet]. Data Migrations. [cited 2015 Feb 22]. Available from: http://www.data-migrations.com/types.html

12. Hemalatha V, Ranjan A. Enhanced Database Migration Technique Using XML. Int J Adv Res Comput Sci. 2013;4(2):248–51.

13. Rouse M. Application migration [Internet]. [cited 2015 Feb 22]. Available from: http://searchcloudapplications.techtarget.com/definition/application-migration

14. Successful Data Migration [Internet]. 2011. Available from: http://www.oracle.com/technetwork/middleware/oedq/successful-data-migration-wp-1555708.pdf

15. Haller K, Matthes F, Schulz C. A detailed process model for large scale data migration projects. In: Business Information Systems [Internet]. Springer; 2012 [cited 2015 Mar 1]. p. 165–176. Available from: http://link.springer.com/chapter/10.1007/978-3-642-30359-3_15

16. Jones D. Is it R.I.P for the Big-Bang Data Migration Strategy? [Internet]. Data Migration Pro. 2012 [cited 2015 Feb 23]. Available from: http://www.datamigrationpro.com/data-migration-articles/2012/11/13/is-it-rip-for-the-big-bang-data-migration.html

17. Data migration strategies [Internet]. [cited 2015 Feb 23]. Available from: http://www.data-migrations.com/strategy.html

18. Jones D. The data migration go-live strategy - what is it and why does it matter? [Internet]. Data Migration Pro. 2009 [cited 2015 Feb 23]. Available from: http://www.datamigrationpro.com/data-migration-articles/2009/3/26/the-data-migration-go-live-strategy-what-is-it-and-why-does.html

19. Walek B, Klimes C. A methodology for data migration between different database management systems. Int J Comput Inf Eng. 2012;(6):80–5.

20. Microsoft Access Data Types [Internet]. MSDN. [cited 2015 Feb 28]. Available from: https://msdn.microsoft.com/en-us/library/ms714540%28v=vs.85%29.aspx

21. Wu B, Lawless D, Bisbal J, Grimson J, Wade V, O'Sullivan D, et al. Legacy system migration: A legacy data migration engine. In: Proceedings of the 17th International Database Conference (DATASEM'97). 1997. p. 129–138.

22. Stonebraker M, Brodie ML. Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach. 1 edition. San Francisco, Calif. : S.l.: Morgan Kaufmann Pub; 1995. 210 p.

23. Brodie ML, Stonebraker M. DARWIN: On the incremental migration of legacy information systems. Distrib Object Comput Group Tech Rep TR-0222-10-92-165 GTE Labs Inc [Internet]. 1993 [cited 2015 Feb 24];56. Available from: http://db.cs.berkeley.edu/papers/S2K-93-25.pdf

24. Tripathy P. Software evolution and maintenance: a practitioner's approach. Hoboken, New Jersey: John Wiley & Sons Inc; 2015.

25. Wu B, Lawless D, Bisbal J, Richardson R, Grimson J, Wade V, et al. The butterfly methodology: A gateway-free approach for migrating legacy information systems. In: Engineering of Complex Computer Systems, 1997 Proceedings, Third IEEE International Conference on [Internet]. IEEE; 1997 [cited 2015 Feb 24]. p. 200–205. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=622311

26. Hollingsworth P. Why Flexibility Is Key to a Successful Data Migration [Internet]. IT Today. 2008 [cited 2016 Feb 3]. Available from: http://www.ittoday.info/Articles/DataMigration.htm

27. Sceales T, Morris J. Four Golden Rules of Successful Data Migration [Internet]. Computing News. 2008 [cited 2016 Feb 2]. Available from: http://home.nestor.minsk.by/computers/news/2008/03/2508.html

28. Informed Consent [Internet]. Meduni Graz. Available from: http://www.medunigraz.at/strategische-projekte/biobank/special-information-for-researchers/ethical-legal-and-social-issues/informed-consent-of-biobank-graz/

29. Pravin, Jain. The class of Java. 1st ed. New Delhi, India: Pearson Education India; 492 p.

30. DuBois P. MySQL [Internet]. 4th ed. Upper Saddle River, NJ: Addison-Wesley; 2009. 1197 p. (Developer's library). Available from: https://books.google.at/books?id=JgF-TUsIC0bUC&printsec=frontcover&dq=mysql&hl=en&sa=X&ved=0ahUKEw-iNz5LF4YTNAhVJxRQKHTheBxwQ6AEIKzAC#v=onepage&q&f=false

31. Introduction to SQL [Internet]. w3schools.com. 2016. Available from: http://www.w3schools.com/sql/sql_intro.asp

32. MySQL [Internet]. MySQL. 2016. Available from: https://www.mysql.com

33. Zeis C, Ruel C, Wessler M. Oracle 11g for dummies. Hoboken, N.J: Wiley; 2009. 392 p. (--For dummies).

34. Greenwald R, Stackowiak R, Stern J. Oracle essentials: Oracle database 11g. 4th ed. Beijing ; Sebastopol, [CA]: O'Reilly; 2008. 386 p.

35. Conrad J. Microsoft Access 2013 inside out. Sebastopol, CA: O'Reilly Media, Inc; 2013. 817 p.

36. Feuerstein S, Pribyl B. Oracle PL/SQL programming. Sixth edition. Beijing ; Cambridge ; Farnham ; Köln ; Sebastopol ; Tokyo: O'Reilly; 2014. 1340 p.

37. McLaughlin M. Oracle database 11g PL/SQL programming. New York: McGraw-Hill; 2008. 835 p.

38. Database PL/SQL User's Guide and Reference [Internet]. Oracle Help Center. Available from: https://docs.oracle.com/cd/B19306_01/appdev.102/b14261/objects.htm

39. Fawcett J, Quin L, Ayres D. Beginning XML. Fifth edition. Indianapolis, Indiana: John Wiley & Sons, Inc; 2012. 827 p. (Wrox beginning guides).

40. Sebestyen TJ. XML: Einstieg für Anspruchsvolle ; [inkl. Lerntest und Beispiele auf CD] [Internet]. München: Addison-Wesley [u.a.]; 2010. 410 p. (Master class). Available from: https://books.google.at/books?id=vvTffWoXkewC&printsec=frontcover&dq=xml&hl=de&sa=X&redir_esc=y#v=onepage&q=xml&f=false

41. Le Hégaret P, Wood L, Robie J. What is the Document Object Model? [Internet]. https://www.w3.org/. 2000. Available from: https://www.w3.org/TR/DOM-Level-2-Core/introduction.html

42. Hunter D. Beginning XML. Indianapolis, IN: Wrox/Wiley Pub.; 2007.

43. Chacon S, Straub B. Pro Git: [everything you need to know about Git]. 2. ed. New York, NY: Apress/Springer; 2014. 426 p. (The expert's voice).

44. Li F, Strickroth S. TortoiseGit Manual [Internet]. TortoiseGit. Available from: https://tortoisegit.org/docs/tortoisegit/index.html

45. Dia Diagram Editor [Internet]. Dia Diagram Editor. Available from: http://dia-installer.de/index.html.en

46. Mindfusion XML viewer [Internet]. Available from: http://mindfusion.eu/xml-viewer.html

47. Solves I. Software Informer [Internet]. Software Informer. 2016. Available from: http://xml-viewer.software.informer.com/

48. Bringing MySQL to the web [Internet]. phpMyAdmin. Available from: https://www.phpmyadmin.net/

49. Notepad++ [Internet]. Notepad++. Available from: https://notepad-plus-plus.org/

50. SQL workbench [Internet]. SQL workbench. Available from: http://www.sql-work-bench.net/index.html

51. Eclipse [Internet]. Eclipse. Available from: http://www.eclipse.org/

52. Teufel M, Helming J. Eclipse 4: Rich Clients mit dem Eclipse 4.2 SDK. Korrigierter Nachdr. Frankfurt am Main: entwickler.press; 2013. 249 p.

53. Collection Strategy [Internet]. Meduni Graz. Available from: http://www.medun-igraz.at/strategische-projekte/biobank/special-information-for-researchers/contents-and-sample-acquisition/collection-strategy-of-biobank-graz

54. Biobank Graz [Internet]. BBMRI. Available from: http://bbmri.at/about-bbmri.at1

55. Biobank Graz [Internet]. Meduni Graz. Available from: https://www.medunigraz.at/strat-egische-projekte/biobank/

56. Gustafsson OJR, Arentz G, Hoffmann P. Proteomic developments in the analysis of for-malin-fixed tissue. Biochim Biophys Acta BBA - Proteins Proteomics. 2015 Jun;1854(6):559–80.

57. Rolls G. Process of Fixation and the Nature of Fixatives. Leica Biosystems.

58. Is Liquid Nitrogen The Only Gold Standard For Long-Term Sample Storage? [Internet]. Thermo Fisher Scientific. 2014. Available from: https://www.thermofisher.com/blog/bi-obanking/is-liquid-nitrogen-the-only-gold-standard-for-long-term-sample-storage/

59. Bassil Y. A comparative study on the performance of the Top DBMS systems. ArXiv Prepr ArXiv12052889 [Internet]. 2012 [cited 2016 Oct 9]; Available from: http://arxiv.org/abs/1205.2889

60. Difference between a Client and a Server [Internet]. TheyDiffer.com. 2016. Available from: http://theydiffer.com/difference-between-a-client-and-a-server/