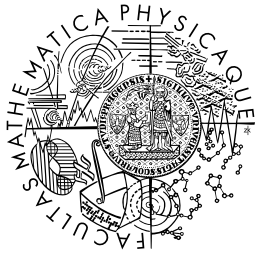

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Denis Drobný

Extracting Information from Database Modeling Tools

Department of Distributed and Dependable Systems

Supervisor of the bachelor thesis:	RNDr. Pavel Parízek, Ph.D.
Study programme:	Computer Science
Study branch:	Programming and Software Systems

Prague 2019

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Dedication.

Title: Extracting Information from Database Modeling Tools

Author: Denis Drobný

Department: Department of Distributed and Dependable Systems

Supervisor: RNDr. Pavel Parízek, Ph.D., Department of Distributed and Dependable Systems

Abstract: Abstract.

Keywords: key words

Contents

1 Introduction **3**
1.1 Goals . . . 5

2 Databases **6**
2.1 Database Types . . . 6
2.1.1 Relational . . . 6
2.1.2 Non-Relational . . . 7
2.1.3 Conclusion . . . 8
2.2 Query Language . . . 8
2.3 Means of Database Access . . . 9

3 Database Modeling **10**
3.1 Abstraction Layer Types . . . 10
3.1.1 Conceptual Data Model . . . 11
3.1.2 Logical Data Model . . . 11
3.1.3 Physical Data Model . . . 12
3.2 Relations Between the Models . . . 12
3.2.1 Maps-to Relation . . . 12
3.3 Construction of a Data Model . . . 12
3.3.1 Modeling . . . 12
3.3.2 Reverse-Engineering . . . 12
3.3.3 Generating . . . 12
3.3.4 Importing . . . 12

4 Modeling Tools **13**
4.1 ER/Studio . . . 13
4.2 PowerDesigner . . . 13

5 Data Lineage **14**
5.1 Theory . . . 14
5.2 Manta Flow . . . 14
5.2.1 Supported Database Technologies . . . 14

6 Analysis & Design of the Solution **15**
6.1 Requirements . . . 15
6.2 Analysis of the Problem . . . 15
6.3 Architecture of the System . . . 15

7 Implementation **16**
7.1 Extensibility . . . 16
7.2 Technologies . . . 16
7.3 Testing . . . 16

Conclusion **17**

Bibliography **18**

List of Figures **19**

List of Tables **20**

List of Abbreviations **21**

A Attachments **22**
A.1 Building 22
A.2 User Documentation 22
A.2.1 Tutorials 22
A.3 Cooperation with Manta Flow 22

1. Introduction

A **database** is a collection of related data. By data, we mean known facts that can be computerized and that have implicit meaning as stated in Fundamentals of Database Systems [1]. We will consider that a database stores data relevant to an enterprise at a host that can be accessed via network. Databases became deep-rooted in every business. Independently of the field that a company is focused on we can enumerate many reasons for considering a database storage deployment a good idea. Let us show some examples of how databases are used through various business domains.

- Social Media
  Every piece of information that has ever been published on social media, from photo through like or comment to friendship establishment, was stored somewhere and that place is a database. Usually the database that a social platform uses does its job in a background. Nevertheless there may occur events when the data storage reminds of its presence as it did on the most recent outage of Facebook. [2].

- Healthcare
  Easy accessibility of large amount of patient's data is a main reason to deploy a database at doctor's office or a healthcare organization [3]. High discretion is a requirement when managing data of such sensitiveness.

- Finances
  complete

- E-commerce
  Every company that sells products online should use a database. The bare minimum is to store offered products themselves and keeping track of purchases that were done by users.

And the list goes on.

A **data model** is a description of data, data relationships, data semantics, and consistency constraints.

A **database schema** defines how is the database described in a data model actually constructed, specifying types of fields from data model. Represents an instance of a data model.

There are multiple kinds of specific data models known. By Fundamentals of Database Systems [4] the main categorization of data models nowadays, according to how big is the abstraction used and on what type of user are aimed, is following:

- Conceptual Data Models (High-Level)
  Reproduces real world objects along with their relationships and should be close to how business end-users perceive them.

- Representational Data Models (Implementation)
  In the middle between the two other model types there are representational data models which on the one hand are comprehensible by end-users and on the other hand are not too abstract so that they can be used as documentation for an actual database implementation of the modeled data.

- Physical Level Data Models(Low-Level)
  In contrast to conceptual models the physical ones are tied with how data are stored physically at storage media showing all specific internal details that may be overwhelming in the case that the reader is a computer specialist.

Once the decision is made and the usefulness of a database for our thing is proved there may be still a long way until everything runs as expected and we can use all the advantages that such data storage brings. The database design phase comes in place then. By the nature of the problem, a top-down approach is usually followed. A database designer comes in handy for the process.

The first thing that is needed to do is to discuss with an expert in the domain what domain in order to identify and collect requirements for the designed system.

The debate should be translated to a conceptual data model of the future database by the designer.

Once the abstract model is created a movement towards its implementation is desired. That is when the development of a logical model takes place and the high level view is turned into system-specific model that is the logical one.

Finally, the organization of the database itself is figured out and captured in a physical model of the analyzed system.

Given the physical model a database should be possible to deploy straightforwardly.

A **diagram** is a graphical visualization of a data model.

A **data modeling tool** is a software that allows a database designer to create data models. End user may use the tools for interactive previewing of the models' diagrams.

Data flow is an internal logic of how data moves in some system across its components.

Data lineage provides a picture of how data moves in some system across its components. It is a description of how data go from an origin through their transformations until they reach a destination. The ability of seeing graphically how data are used, what for, and what are the consequences of the usage in a system is a powerful tool for error tracing.

The process of development and deployment of a database consists of multiple stages as we have seen. At the beginning there is a high level view of why the database is needed and what purpose will it serve. Hopefully, in some time the result is that the data described in the initial step are stored physically at some server. This way the data can be accessed and processed. What we want to achieve in this work is to make use of the individual steps taken during the design process, and make operations on data as transparent and traceable as possible even for business users that don't have a technical background.

1.1 Goals

- Develop a component that extracts metadata from database models that were created using SAP PowerDesigner
- Develop a component that extracts metadata from database models that were created using ER/Studio
- Provide a description by means of a programming language for a general scenario of metadata extraction from a data modeling tool output and passing the information to a data lineage tool
- Propagate data lineage acquired by analysis of how is database used and constructed to more abstract data models than is the physical one, to the logical and the conceptual models.

Introduction to each of the following chapters once the final organization is known

2. Databases

A standalone database is not very useful. To take the full advantage of it we need some means to define, create, maintain and control access to the database. That is purpose of a software called **Database Management System (DBMS)**.

2.1 Database Types

We already described why we want to use a database and roughly mentioned what are the pieces of data that we want to save there. Now let's take a look at what are differences between in database implementations and what to take in account when comparing database technologies. That may be helpful when choosing the best suitable option for some specific data set to store or to see how storing of great amount of structured information can be approached.

The basic division of databases types is simple and binary - they are either Relational or Non-Relational.

There are Database Management Systems build around both, Relational Database Management System (RDBMS)

2.1.1 Relational

A **Relational Database** is a set of tables. A table consists of rows (also records) and columns. We can see such table as an object whose attributes are represented by columns and instances by rows. The important thing is that relational tables tables carry both data that need to be stored by user and the relationships between the data as well. To store an atomic piece of data about instance a proper column is filled with a value. Whereas to capture a relationship between objects the concept of keys is used.
A **Key** is a subset of table's columns used for identifying a record.
A **Primary Key** is a Key that non-ambiguously identifies a record in table and is used when referring to the record.
A **Foreign Key** is a Key that uniquely identifies a record from a table (may be the same or a different one).
They are known also as SQL databases by the query language used in RDBMS for managing data.

The Most Used Relational Database Management Systems [1]

- Oracle
- MySQL
- Microsoft SQL Server
- PostgreSQL

[1] The database technologies usage statistics are based on data from the most up to date version of website db-engines.com [5].

- IBM Db2

Advantages [6] [7]

- Designed for managing structured data
- ACID compliance - database transactions are Atomic, Consistent, Isolated, Durable
- The technology is mature, well-established with large ecosystem and many developers have experiences with SQL and RDBMS
- Data integrity is enforced

Disadvantages

- Problems managing data that are unstructured or semi-structured
- Data is normalized in order to achieve reduction of data redundancy, therefore stored objects may not have one-to-one mapping with the tables that represent them in memory. Also meaning lots of expensive (in terms of speed) joins when fetching such objects.

2.1.2 Non-Relational

A **Non-Relational Database**, is any database that does not follow the relational paradigm. They are younger and were invented to overcome limitations that relational engines have. The ultimate aim is to be more effective when coping with Big Data - data that is fast growing and their structure may not be defined strictly (unstructured, semi-structured information) [8]. There are multiple ways that these requirements can be met so we will introduce more precise division [9]. They are also commonly referred as NoSQL databases as the opposite of SQL databases.

Detailed Description of the types?

Non-Relational Database Types The most used DBMS is listed with each type.

- Key-value stores [10]
  Redis
- Wide column stores
  Cassandra
- Document stores
  MongoDB
- Graph databases
  Neo4j
- Search engines
  Elasticsearch

Advantages [6] [7]

- Elastic scaling, new cluster nodes can be added easily
- No strict database schema is required, bigger flexibility when changing inserted data format

Disadvantages

- Weaker data consistency mode - BASE (Basically Available, Soft state, Eventual consistency) in contrast to stronger ACID in RDBMS
- Lack of built-in data integrity
- Join operation is hard and may be even not supported

2.1.3 Conclusion

By the described properties of the respective systems, hopefully, a reader has some image of in what situation is reasonable to use the more traditional Relational design or look around for one of the Non-Relational databases. To sum it up, if the ACID principle is required by a user and business rules should be enforced the SQL databases are the ones to choose. Enterprises should be cautious and their first choice would be a Relational Database. In contrast when storing heterogeneous data or big volumes of it, consistency is not a priority and the system is extensively distributed some of the Non-Relational databases may be the right one.

Move to the modeling chapter?

However, in this work we will focus only on databases that are of the Relational kind.

The main reason behind this is that since NoSQL Databases have flexible schema or are schema-less (there is no point in determining a database schema when data types of attributes or keys) modeling of these databases quite a new discipline and is hard to find an intersection among different approaches to NoSQL modeling. Also concepts of higher abstraction models are omitted. [11]

The second thing to consider is that once a database is Relational we more or less know what to expect from it. The structure of these databases . So a tool that would extract metadata from relational data models is potentially more powerful as it can be applied to more database technologies than a similar tool aimed for some specific type of Non-Relational database.

Lastly, despite the Non-Relational may be growing in numbers and becoming a serious player, as it suits some use-cases better, the Relational still are, and in the near future will be, far more widely used.

2.2 Query Language

?

2.3 Means of Database Access

We have a database that stores some data, the data may be queried and modified via SQL statement in Database Management System. However this can be insufficient as third party programs, let's call them application programs, would want to access the DBMS. A solution is to provide them with an application programming interface (API) that provides a set of methods available in the programming language that the application program was written in, so it can use them. Most commonly when the API is called its implementation translates the request so that to a specific DBMS driver that it is passed after understands it and performs the desired action.

A **Connection String** is a textual information used to identify a data source and establish a connection with it. It it is made of pairs of keywords and values separated by semicolon, the keywords are parameters of the connection.

APIs to DBMS

- Open Database Connectivity (ODBC)
  General, language independent
- Java Database Connectivity (JDBC)
  The Java ecosystem
- ADO.NET
  .NET Framework

3. Database Modeling

The first question that has to be answered is what does data modeling brings us.

One may ask why it is necessary to develop some models before an actual database creation. But let us imagine building a house without solid design and documentation. It sounds a bit strange to hire construction workers straight ahead and tell them that we need a house that has 5 rooms, some toilets and expect a good result. Most probably some building would be produced, but we will agree that expectations and requirements of the later inhabitant could not be met properly. Surely there are good reasons why the usual steps are followed strictly. Let us move on from the analogy to the database domain.
When deploying a database from a scratch we may think of two short term advantages. Firstly, the time needed to have data stored somewhere would be much shorter and secondly the initial cost of the system could be lower.
But over time both of the advantages will, most likely, get outnumbered by problems that will begin to appear. Maintenance of a poorly designed system (or not designed at all) is expansive and leads to numerous outages.

Data modeling should lead to higher quality as is pushes to thorough definition of the modeled problem. Once we know what to solve and what is the scope, it is much easier to come with different solutions and justify which of the proposed approaches is the most suitable one.

Costs are reduced since during creation of a data model many errors are identified thus can be caught early, when they are easy to fix.

Data models form a nice piece of documentation that is understandable by each of the involved parties. When someone tries to understand the system, he can choose a data model on an appropriate level of abstraction that will introduce him the important aspects of the problem that suits his knowledge and qualification.

Also during the design process we may learn a lot about properties of the data that we need or have and will be stored. These information are crucial for choosing an appropriate type of database, whether to stick with a relational database if so which DBMS is the one for us, or to look for a non-relational one.

3.1 Abstraction Layer Types

Some time ago, in 1975, American National Standards Institute [12] first came with database architecture consisting of three levels:

- External Level
- Conceptual Level
- Physical Level

Now let's have a look at what different options are available when creating a data model of a database. The first categorization of these models was proposed 1 - conceptual (also high-Level), logical (also representational, or implementation) and physical (also low-level) data models.

3.1.1 Conceptual Data Model

The purpose of a conceptual data model is to project to the model real-world and business concepts or objects.

Characteristics

Aimed to be readable and understandable by everyone.
Is completely independent of technicalities like a software used to manage the data, DBMS, data types etc.
Is not normalized.

A real world object is captured by an **entity** in conceptual model if our modeling domain is public transport then entity may be a bus or a tram. For further description of objects that we are interested in **attributes** are used, those are properties of entities, for example a license plate number would an information to store when describing buses. Also **relationships** between objects are necessary to provide full view of the section of the world that a data model resembles. Having transportation companies in our data model it is really fundamental to see that a company may own some vehicles.

Types of data models used for CDM

But if someone speaks about a conceptual data model most probably he means an entity-relationship data model.

3.1.2 Logical Data Model

Keeping its structure generic a logical model extends the objects described in a conceptual data model making it not that easy to read but becomes a good base documentation for an implementation.

Characteristics

Independent of a software used to manage the data or DBMS.
Data types description is introduced (but in a way that is not tied with any specific technology).
Normalized up to third normal form.

3.1.3 Physical Data Model

Characteristics

3.2 Relations Between the Models

3.2.1 Maps-to Relation

3.3 Construction of a Data Model

3.3.1 Modeling

Each layer or not

3.3.2 Reverse-Engineering

Physical layer

3.3.3 Generating

Downwards

3.3.4 Importing

Each layer

4. Modeling Tools

4.1 ER/Studio

4.2 PowerDesigner

5. Data Lineage

5.1 Theory

5.2 Manta Flow

5.2.1 Supported Database Technologies

6. Analysis & Design of the Solution

6.1 Requirements

6.2 Analysis of the Problem

6.3 Architecture of the System

7. Implementation

7.1 Extensibility

Description of the common structure of the solution - what to do when a programmer wants to write a connector for another modeling tool.

7.2 Technologies

7.3 Testing

Conclusion

Bibliography

[1] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems (7th Edition)*. Pearson, 2015.

[2] Facebook blames 'database overload' for most severe outage in its history as users continue to report problems with Instagram and WhatsApp more than 14 hours after global meltdown.

[3] HealthCare.gov's Heart Beats For NoSQL.

[4] Abraham Silberschatz Professor, Henry F. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill Education, 2010.

[5] DB-Engines Ranking.

[6]

[7]

[8] What Is A Non Relational Database.

[9] The Types of Modern Databases.

[10] What Is a Key-Value Database?

[11]

[12] SPARC-DBMS Study Group. Interim Report: ANSI/x3/SPARC Study Group on Data Base Management Systems, 1975.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 Building

A.2 User Documentation

A.2.1 Tutorials

A.3 Cooperation with Manta Flow