



F3

**Faculty of Electrical Engineering
Department of Cybernetics**

Bachelor's Thesis

Automatic Evaluation of an ER Diagram Transformation into a Relational Model

Sára Krinerová
Open Informatics

May 2024
Supervisor: RNDr. Ingrid Nagyova, Ph.D.

I. Personal and study details

Student's name: **Krinerová Sára** Personal ID number: **499317**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Cybernetics**
Study program: **Open Informatics**
Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Automatic Evaluation of an ER Diagram Transformation into a Relational Model

Bachelor's thesis title in Czech:

Automatická evaluace transformace ER diagramu do relačního modelu

Guidelines:

The aim of the work is to design and implement a system for automatic checking of the tasks in the Database Systems course. The purpose of the task is to verify students' ability to correctly transform an ER diagram into a relational model. The system will be evaluated on tasks submitted to BRUTE.

1. Specify the roles for the transformation of an ER diagram into a relational model. Analyze the possible variability of the conversion and describe how the resulting relational model may be different for a given ER diagram.
2. Familiarize yourself with the data format in which assignments are submitted in BRUTE. Specify the possibilities of computer processing of these data and, if necessary, suggest alternative ways of submitting the assignments.
3. Design and implement a system for checking the corresponding task.
4. Perform a system testing on the tasks submitted to BRUTE and compare the results with the results of the evaluation of the subject teachers.

Bibliography / sources:

- [1] Svoboda, M. „Lectures from the Database Systems course.“ 2021. Available at: <https://www.ksi.mff.cuni.cz/~svoboda/courses/182-B0B36DBS/> [Accessed 31st January 2024].
- [2] Pokorný J. - Valenta M. „Databázové systémy.“ Nakladatelství VUT, Praha, 2013.
- [3] Codd, E. F. A „Relational Model of Data for Large Shared Data Banks.“ Reprinted from Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387.
- [4] „Database SQL Reference.“ Oracle Documentation. 2015. Available at: https://docs.oracle.com/cd/B19306_01/server.102/b14200/toc.htm [Accessed 31st January 2024].
- [5] Škarda, A. „Automatické vyhodnocování úloh v p edm tu Databázové systémy.“ Bachelor thesis, VUT FEL. 2023.

Name and workplace of bachelor's thesis supervisor:

RNDr. Ingrid Nagyová, Ph.D. Center for Software Training FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **06.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

RNDr. Ingrid Nagyová, Ph.D.
Supervisor's signature

prof. Dr. Ing. Jan Kybic
Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Acknowledgement / Declaration

I would like to thank my family not only for their financial support but also for keeping me well supplied with sweets. I would also like to thank Max Hollmann for not taking all the sweets and for being there for me whenever I needed emotional support.

My great thanks go to Sabina Jaroušková for her language proof-reading.

Lastly, I would like to thank my supervisor, RNDr. Ingrid Nagyová, Ph.D., for her patience, time, and assistance during the writing of this thesis.

I declare that I have completed the submitted work independently and that I have cited all the sources of information used in accordance with the Methodological Instructions for observing to Ethical Principles in the Preparation of University Theses.

In Prague on 24th May 2024

.....
Sára Krinerová

Abstrakt / Abstract

Tato práce se zaměřuje na vyhodnocení semestrální práce v předmětu Databázové systémy, kde je cílem převést ER diagram na relační model. První část je věnována popisu problému, definici ER diagramu, relačního modelu a prezentaci všech pravidel pro převod z ER diagramu do relačního modelu. Druhá část práce se soustředí na analýzu prací studentu. V poslední části je popsána implementace včetně výsledků a závěru.

Klíčová slova: Vyhodnocení úkolu, ER diagram, Relační model, Konceptuální modelování, Databázové systémy

This work focuses on evaluating the semester assignment in the Database Systems course, where the goal is to convert an ER diagram into a relational model. The first part is dedicated to describing the problem, defining ER diagrams, the relational model, and presenting all the rules for the conversion from an ER diagram to a relational model. The second part of the work focuses on analyzing student solutions and designing the system. In the final section is described the implementation, including the results and conclusion.

Keywords: Task Evaluation, ER Diagram, Relational Model, Conceptual Modeling, Database Systems

Contents /

1 Introduction	1		
1.1 Objectives	1		
1.2 Structure of the thesis	1		
2 ER diagram and Relation Model	2		
2.1 ER Diagram	2		
2.1.1 Components of the ER Diagram	2		
2.1.2 Table of Components	3		
2.2 Relation model	4		
2.2.1 Scheme notation	4		
2.2.2 Identification	4		
2.3 Transformation from ER diagram to relational model	5		
3 Analysis of student mistakes	11		
3.1 Types of mistakes	11		
3.2 The frequency of mistakes	12		
3.3 Summary	14		
4 Analysis of inputs	15		
4.1 Input from students	15		
4.2 Analysis of PDF	15		
4.2.1 Libraries for reading PDF	15		
4.2.2 PyPDF2	16		
4.2.3 Pdftminer	16		
4.2.4 PyMuPDF	17		
4.2.5 PDFAnnots	17		
4.2.6 Summary	17		
4.3 Reading the underline text in PDF	18		
4.3.1 PDF to DOC/X	18		
4.3.2 PDF to HTML	18		
4.4 Suggestion for changing the assignment	19		
4.5 The input from the Con- ceptual Model task	19		
5 Project requirements	21		
5.1 Requiremenets	21		
5.2 Design	22		
5.3 Technical design	22		
6 Implementation	24		
6.1 Steps of implementation	24		
6.1.1 Classes	24		
6.1.2 Parsing input CP-1	25		
6.1.3 Parsing problems CP-1 input	25		
6.1.4 Parsing the CP-2 input	26		
6.1.5 Problems with Parsing the CP-2 Input	27		
6.1.6 Checking	28		
6.1.7 Problem with Checking	28		
6.1.8 Final Points	28		
6.1.9 Printing the output	28		
6.1.10 Configuration	29		
6.1.11 Program parameters	29		
6.2 Structure of the program	29		
6.3 Timeline of the project	30		
7 Testing and results	33		
7.1 Testing	33		
7.2 Results	33		
7.2.1 Types of mistakes	34		
7.3 Summary	35		
8 Conclusion	36		
References	37		
A Glossary	39		
B Attachments	40		
B.1 Source coude	40		
C Template	41		

Chapter 1

Introduction

Within the Database Systems course, students undertake a sequence of homework assignments that lead to the creation of a database. For many years, the individual parts of these assignments were graded manually, which took a considerable amount of time for the teachers. As the number of students continues to grow each year, the workload of manually grading these assignments has also increased.

1.1 Objectives

The thesis aims to analyze the problem and design and implement a program for evaluating students' ability to transform ER diagrams into relational models, which is one part of the semester project. The system will be evaluated on tasks submitted to BRUTE, the platform used for assignment submissions in the Database Systems course.

1.2 Structure of the thesis

This thesis is organized into several chapters, each addressing a specific aspect of the project. Chapter 2 defines ER diagrams and their components, organized in a table, it continues with the definition of relational models and their notation. The final part of this chapter discusses the transformation from ER diagrams to relational models. Chapter 3 analyzes common mistakes made by students during the transformation process, categorizing the types of mistakes and their frequencies. Chapter 4 examines the inputs that the program will process and explores possible solutions for reading and working with these inputs. Chapter 5 reiterates our goals and outlines the project requirements, covering both functional and technical design specifications. Chapter 6 describes the implementation steps of the system, explaining the structure of the program and the challenges encountered during development with solutions. Chapter 7 presents the results of system testing, providing a comparison with manual evaluation results. Finally, Chapter 8 concludes the thesis with a summary of findings, implications, and suggestions for future work. Additionally, the thesis includes a glossary and attachments that provide additional resources, terminology definitions, source code, and templates.

Chapter 2

ER diagram and Relation Model

This chapter is focused on the definition of the Entity Relationship (ER) diagram, the relational model, and the conversion from the ER diagram to the relational model. It outlines all the rules and conversion options that will be used in the implementation part.

2.1 ER Diagram

According to Gavin Powell [1] *Entity Relationship diagram is a diagram that represents the structural contents in tables for an entire schema in a database.* In ER diagram is included schematic representations of relationship between entities, represented by various types of relationship, primary keys and foreign keys[2].

2.1.1 Components of the ER Diagram

The ER diagram is composed of several components [3]. Here are the basic ones:

- Entities
 - Strong Entity
 - Weak entity
- Attributes
 - Simple attribute
 - Key attribute
 - Composite attribute
 - Composite key attribute
 - Single-valued attribute
 - Multi-valued attribute
- Relationship
 - One-to-One relationships
 - One-to-Many relationships
 - Many-to-One relationships
 - Many-to-Many relationships
- Other
 - ISA hierarchy
 - Recursive relationship

2.1.2 Table of Components

There are many ways in which individual components can be represented graphically. Among the most famous are, for example, Chen Notation, Bachman Notation or IDEF1X Information Model Notation, the advantages of which are described together with other notations in the article A Comparative Analysis of Entity-Relationship Diagrams [4]. Within the Database Systems course, the ERDIA[5] web application is used, where the individual components are represented as shown in the 2.2.

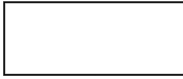
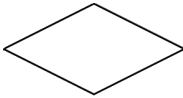
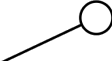
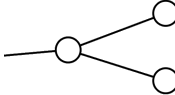
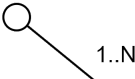
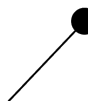
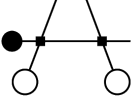
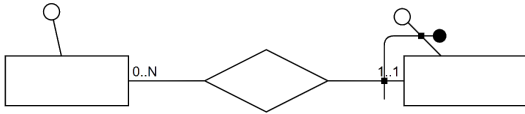
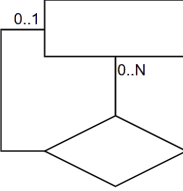
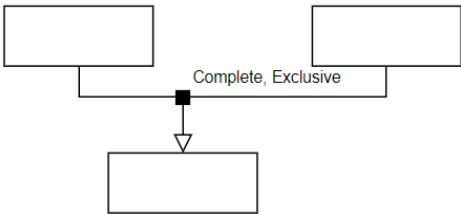
Name	Picture
Entity	
Relationship	
Attribute	
Composite attribute	
Multi-valued attribute	
Identifier	
Composite identifier	
Weak entity	
Recursive relationship	
ISA hierarchy	

Table 2.1. Components of the ER diagram

2.2 Relation model

A relational model describes a structure for storing and manipulating information in a database. This idea was introduced by E. F. Codd in 1970 [6] and thus preceded the existence of the ER diagram.

2.2.1 Scheme notation

Scheme notation is a description of a relational structure [7].

$$S(A_1:T_1, A_2:T_2, \dots, A_n:T_n)$$

- S is a schema name, thus the name that describes the table for example 'Person'.
- A_i are attribute names and T_i their types (attribute domains)
- Specification of types is often omitted

Example:

```
Person(personalId:Integer, firstName:String, lastName:String)
```

Types are usually not specified because they are evident from the context.

2.2.2 Identification

In the relational model is each tuple identified by one or more attributes. The significance of these identifiers lies primarily in unambiguously determining and facilitating easier access to a given tuple. This means that every table must contain at least one identifier [8].

- Superkey: A superkey is a set of attributes that uniquely identifies each tuple. That means that we don't need a key, but all attributes together form a single superkey
- Primary key: A primary key is a minimal set of attributes that uniquely specify each tuple in a relational table. This means that no attribute can be removed from the key without losing the uniqueness of the tuple. However, each tuple can contain multiple keys that sufficiently identify it and each of the keys may have a different number of attributes.
- Foreign key: A set of attributes that refers to another table, where it corresponds to the (super)key of the referenced relationship. It is usually not a (super) key in the referenced table, but it can be or be part of it.

Example:

```
Person(personalId, firstName, lastName, address)
```

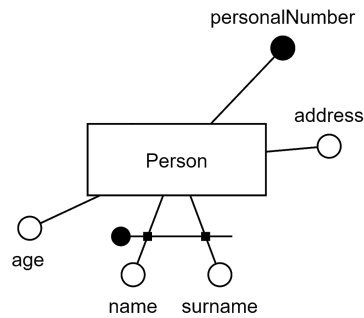
```
Pet(petId, name, person)
```

```
FK:(person) ⊆ Person(personalId)
```

2.3 Transformation from ER diagram to relational model

In this section, we will go through all the rules for transforming an ER diagram into a relational model. We will describe them for individual cases and also show additional transformation options.

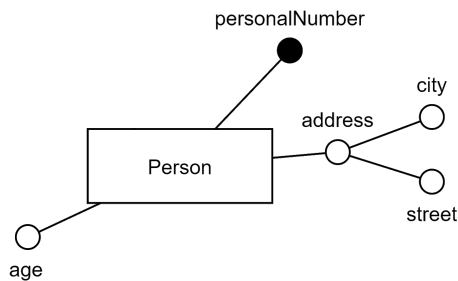
Rule 2.1. Entity



Person(personalNumber, name, surname, address, age)

In the relational model, the entity is represented by the table with corresponding attributes. The key attributes are underlined and all attributes in the composite key is underlined with one line.

Rule 2.2. Composite attribute



Option n.1:

Person(personalNumber, age)
 Address(person, street, city)
 FK:(person) \subseteq Person(personalNumber)

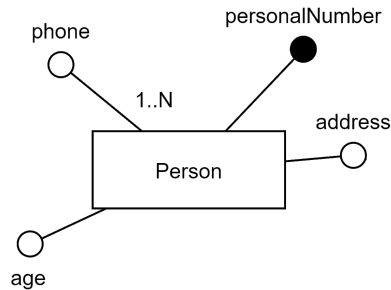
Option n.2:

Person(personalNumber, street, city, age)

Composite attribute in relational model has two possible representation. Typically, the individual attributes of a composite attribute are present in the original table. In cases where access to individual attributes of a composite attribute is rarely or

infrequent or they are not needed in the original table, it is possible to create a new table for the composite attribute. The new table is identified by the key from the original table and this key is also has to be marked as foreign key.

Rule 2.3. Multi-valued attribute



Cardinality (0..N) or (1..N):

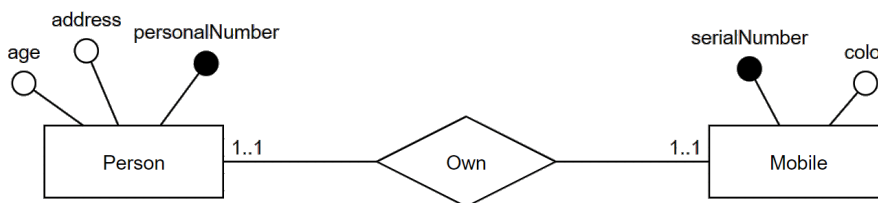
Person(personalNumber, address, age)
 Phone(person, phone)
 FK:(person) \subseteq Person(personalNumber)

Cardinality (0..1):

Person(personalNumber, address, age)
 Phone(person, phone)
 FK:(person) \subseteq Person(personalNumber)

In the relational model, a multi-valued attribute is always represented as a separate table, where the composition of the table depends on the cardinality. For cardinality (1..N) or (0..N), the table has only one composite key consisting of the attribute itself and the key from the original table. For cardinality (0..1), the table has only key from the original table and the given multi-valued attribute as a normal attribute.

Rule 2.4. Multiplicity relationship (1..1):(1..1)



Option n.1:

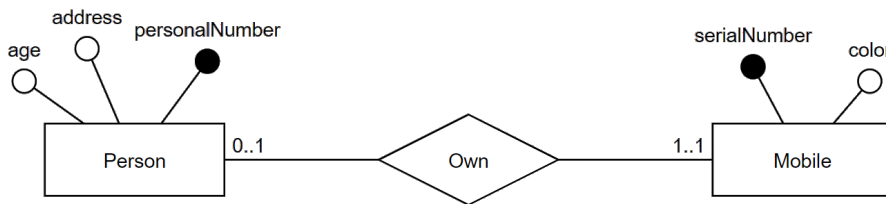
Person(personalNumber, address, age)
 Mobile(serialNumber, color)
 Ownership(person, mobile)
 FK:(person) \subseteq Person(personalNumber)
 FK:(mobile) \subseteq Mobile(serialNumber)

Option n.2:

Person(personalNumber, address, age, serialNumber, color)

We have two options for transforming relationship (1..1):(1..1) to relational model. We can either create a new table that will contain foreign keys from both original tables, and each foreign key would be marked as a primary key because both tables are related exactly once. Or, if we need to access data from both tables concurrently often, we can create a single table that would contain keys and attributes from both tables.

Rule 2.5. Multiplicity relationship (0..1):(1..1)



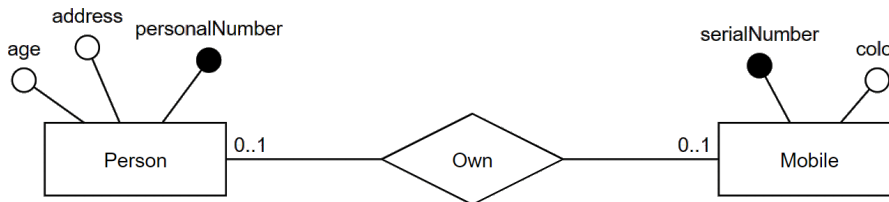
Person(personalNumber, address, age)

Mobile(serialNumber, color, person)

FK:(person) \subseteq Person(personalNumber)

The relationship between two entities, where one table has an obligation to be in relationship and the other table may or may not have, but at most once, is transformed into a relational model as two tables. The table that has the obligation to be in the relationship would apart from the second table remains the same. In addition to its keys and attributes, the second table will also contain a foreign key from the first table, which will be marked as the primary key - thus, the relationship ensures that it will always be unique. If we were to combine both tables together, we would not avoid NULL values.

Rule 2.6. Multiplicity relationship (0..1):(0..1)



Person(personalNumber, address, age)

Mobile(serialNumber, color)

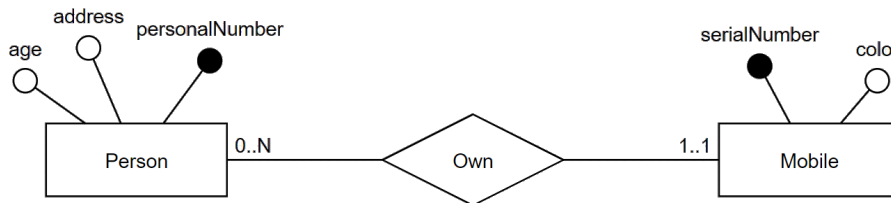
Ownership(person, mobile)

FK:(person) \subseteq Person(personalNumber)

FK:(mobile) \subseteq Mobile(serialNumber)

A relationship of (0..1):(0..1) is transformed into the relational model as a new table. Because tables from entities can exist independently of each other and we want to avoid NULL values, we need to create a new table, formed by foreign keys from both tables. Since the relationship between the two tables can be at most one, these foreign keys would be in the new table each marked as the primary keys.

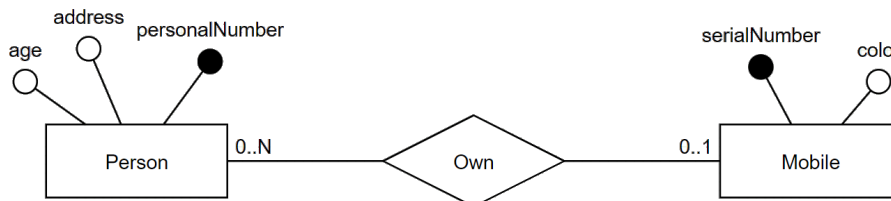
Rule 2.7. Multiplicity relationship (0..N)/(1..N):(1..1)



Person(personalNumber, address, age)
 Mobile(serialNumber, color, person)
 FK:(person) \subseteq Person(personalNumber)

In this relationship, one table has the obligation to be in the relationship, while the other can be multiple times. The table that can be multiple times will remain the same, while the table that has the obligation to be in the relationship will contain a foreign key from the first table. This foreign key will be present in the table as an attribute because there can be multiple tables with this foreign key in the relationship, so it cannot serve as the primary key.

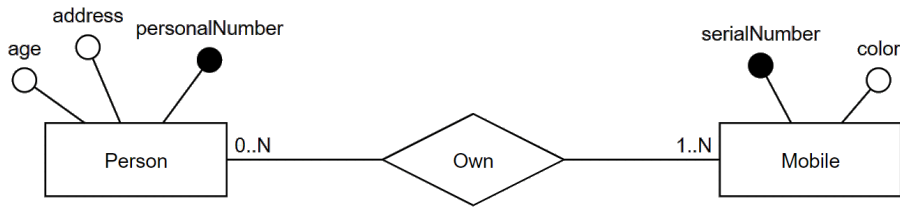
Rule 2.8. Multiplicity relationship (0..N)/(1..N):(0..1)



Person(personalNumber, address, age)
 Mobile(serialNumber, color)
 Ownership(person, mobile)
 FK:(person) \subseteq Person(personalNumber)
 FK:(mobile) \subseteq Mobile(serialNumber)

For this relationship, we need to create three tables, where the first two will correspond to each of the entities. The third table will be formed by foreign keys from the previous two tables. However, in this table, the key will only be the one corresponding to the relationship (0,1), because there cannot be more tuples than one containing this foreign key. The second foreign key may be in multiple tuples, so it is not a key in the third table.

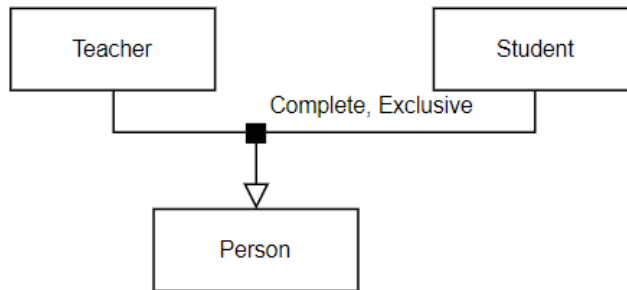
Rule 2.9. Multiplicity relationship (0..N)/(1..N):(0..N)/(1..N)



Person(personalNumber, address, age)
 Mobile(serialNumber, color)
 Ownership(person, mobile)
 FK:(person) ⊆ Person(personalNumber)
 FK:(mobile) ⊆ Mobile(serialNumber)

This relationship is transformed into the relational model as three tables, where the first two tables again correspond to each of the two entities and thus contain the same attributes, including the key. The third table is formed by foreign keys from the first two tables, and these foreign keys together form a composite key for the third table. The reason is that individual foreign keys can appear in N tuples, and we have uniqueness ensured only by their combination.

Rule 2.10. ISA hierarchy



Exclusive:

Person(personalNumber, address, age)
 Teacher(person, address, age, mobile)
 Student(person, address, age, specialization)
 FK:(person) ⊆ Person(personalNumber)
 FK:(person) ⊆ Person(personalNumber)

Overlapping:

Person(personalNumber, address, age)
 Teacher(person, department, mobile)
 Student(person, specialization)
 FK:(person) ⊆ Person(personalNumber)
 FK:(person) ⊆ Person(personalNumber)

There are two types of ISA hierarchies: overlapping and exclusive. For both options, descendants inherit at least the primary key from the parent, which must be marked as a foreign key to the parent table. In overlapping inheritance, descendants may inherit only the key from the parent and should also contain their own non-inherited attributes. In exclusive inheritance, besides inheriting the key from the parent, the child also inherits all attributes and may have additional attributes. All inherited keys and attributes must then be marked as foreign keys to the parent table.

Other relationships (e.g.recursive) are defined according to the specified Rules 2.1-2.10..

Chapter 3

Analysis of student mistakes

In this chapter, we will focus on analysis of most common mistakes made by students. Problem analysis, in our case the most common mistakes, is an important part of the implementation design because it is as an insight into the problem and highlights what is important to focus on and not forget during the program implementation.

First we will list the mistakes identified by teachers, we will examine the frequency of individual mistakes identified by teachers. Later, we will compare these mistakes with the mistakes found by our evaluation program.

3.1 Types of mistakes

In this section, we will list all types of mistakes found by teachers in the Database Systems course in the summer semester of 2022

We will label each mistake as Axx, where xx corresponds to the number in the order.

- A01 - Additional attribute
- A02 - Missing attribute
- A03 - Additional primary key
- A04 - Missing primary key
- A05 - Has more primary keys instead of one composite key
- A06 - Has one composite key instead of multiple primary keys
- A07 - Additional attribute in the composite key
- A08 - Missing attribute in the composite key
- A09 - Additional table
- A10 - Missing table
- A11 - Missing relationship
- A12 - Incorrect relationship
- A13 - Additional foreign key
- A14 - Missing foreign key
- A15 - Composite foreign key in a new table as a single attribute
- A16 - Multiple foreign keys as FKs for the same table
- A17 - Foreign key refers to a table where this attribute does not exist

- A18 - One composite foreign key split into two
- A19 - Different name than in the ER diagram
- A20 - Same attribute repeated multiple times in one table
- A21 - Incorrect recursive relationship
- A22 - Missing recursive relationship
- A23 - Missing SET
- A24 - Missing multivalued attributes
- A25 - Missing hierarchy

To clarify some mistakes, we will illustrate them with examples:

Example:

Roaster(name, address)

Provides(roaster, cafe, cafe-address)

FK: (roaster) \subseteq Roaster(name)

FK: (cafe, cafe-address) \subseteq Cafe(name, address)

- A15 - The foreign key refers to multiple attributes, but in the new table, there is only one. This means that we are concatenating multiple columns together.

Example:

FK: (cafe) \subseteq Cafe(name, address)

- A16 - The same table refers to another table with multiple foreign keys, although only one is needed. Additional keys in this case are redundant and in some cases, they may even alter the composite key.

Example:

Provides(roaster, cafe, cafe-name, cafe-address)

FK: (roaster) \subseteq Roaster(name)

FK: (cafe) \subseteq Cafe(id)

FK: (cafe-name, cafe-address) \subseteq Cafe(name, address)

- A20 - One table contains duplicate instances of the same attribute, and sometimes one of them may be marked as a key.

Example:

Cafe(id, name, address, name, owner)

3.2 The frequency of mistakes

In this section, we will examine the distribution of the listed mistakes, which mistake are most common among students, and which ones are rare.

For our analysis, data from 2 parallel groups totaling 42 students were provided. Given that it is more important for us to identify which mistakes are common among students and we expect that each student may repeat the same mistake, we'll count each type of mistake only once, even if it occurred multiple times for a student.

The first graph represents the frequency of individual mistakes compared to the total number of mistakes, with the numbering corresponding to the numbering in

the types of mistakes 3.1.

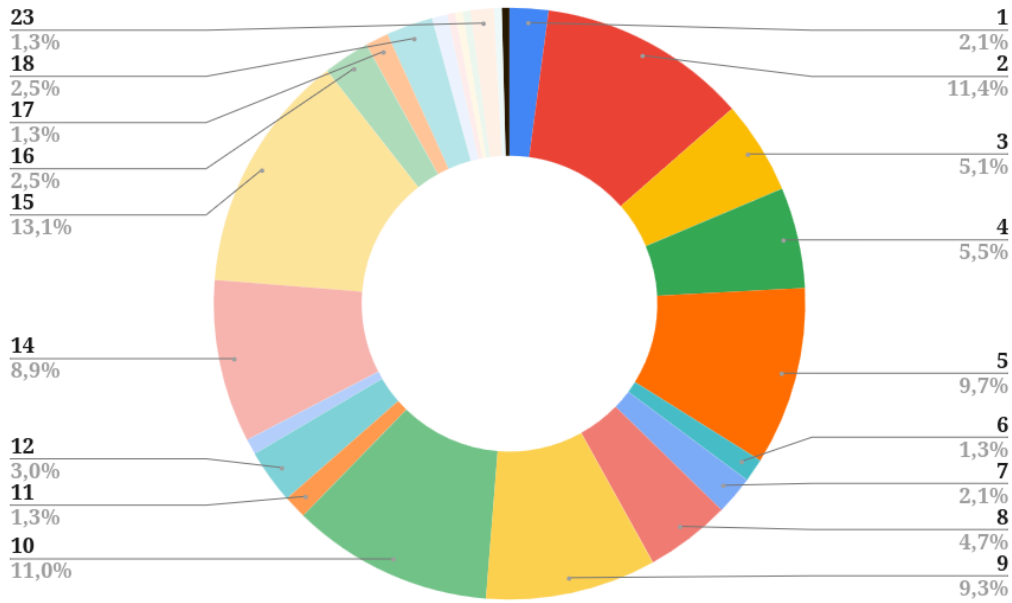


Figure 3.1. Graph of the most common mistakes made by students

The second graph illustrates the number of students who made each mistake. This graph is primarily to give us an idea of how often each mistake occurs among students and we'll use it later when comparing it with the number of mistakes found by the evaluation program.

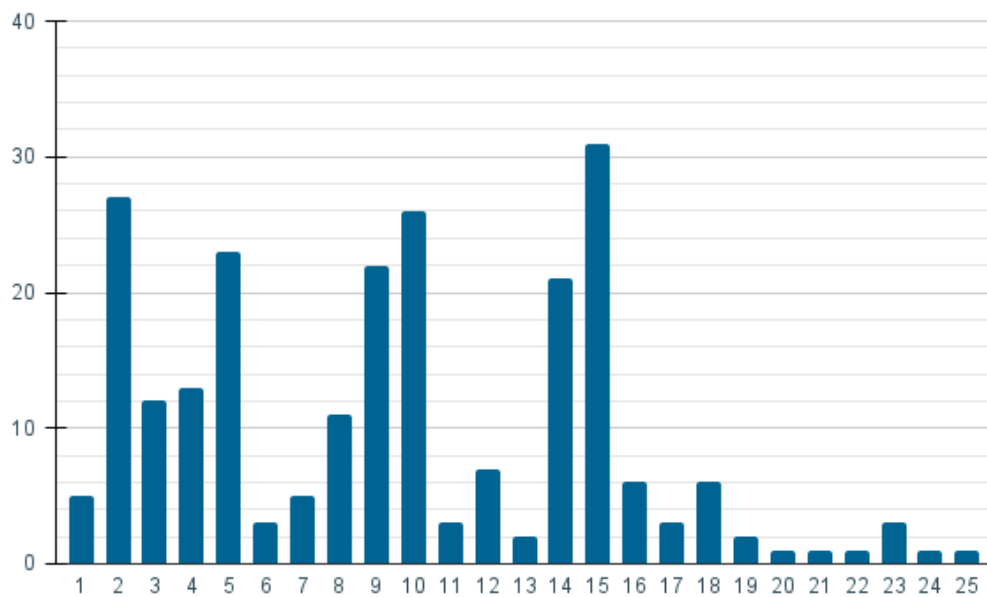


Figure 3.2. Graph of count of mistakes per student

3.3 Summary

The mistakes analysis helped us identify the most common mistakes among students.

The biggest challenge for students is mistake A15, where a composite attribute is incorrectly treated as a single attribute in a foreign key. This mistake accounts for 13% of all mistakes and was found in approximately 3/4 of the students. Together with mistakes A2, A5, A9, A10, and A14, they make up almost 2/3 of all mistakes, representing a significant portion of the overall mistake rate. Unfortunately, for some of these mistakes, we are unsure of their root cause. For instance, mistake A10 (Missing table) could be caused by a missing relationship, a multi-valued attribute, or even forgetting to convert an entity altogether. It's crucial for us to handle all these cases properly and attempt to differentiate their origins.

Moreover, thanks to the analysis of student mistakes, we discovered mistakes that we had not considered before, such as duplicate attributes in a single table. This gave us a better understanding of what to expect in the student input.

Chapter 4

Analysis of inputs

In this chapter, we will go through individual inputs, analyze them, and their reading. This step is essential for implementing input reading because without correctly read input, we will not be able to correct anything properly.

4.1 Input from students

The first input we will be reading is the second part of the semester project in Database systems course, the Relational model (CP-2), from the student. Our goal is to check and evaluate the student's assignment, so it is very important to be able to read this input correctly.

For the summer semester 2022/23¹, this task was assigned as follows:

- Transform the conceptual model(CP-1) to relational model in form, submit in the PDF document form containing an image of the conceptual model(corrected if needed) and the relational model
- If you consider it appropriate, enhance the previous conceptual model with new entities/relationships/attributes.
- Use textual notation $Table(\underline{Key}, Attribute1, Attribute2)$, specify foreign keys. Do not consider NULL values.

4.2 Analysis of PDF

The assignment requires submitting this task in PDF format, which means we will focus first on PDFs and reading text from them.

Adobe's Portable Document Format (PDF) is a very useful document format; it is independent of the operating system and can be viewed on any computer [9].

In the PDF documentation [10], we find that PDF was developed and specified by Adobe Systems Incorporated beginning in 1993 and continuing until 2007. This means that it is not a new format; on the contrary, it was chosen for the Database Systems course because of its widespread use and availability.

Nevertheless, each PDF generator differs in how it writes individual elements, which we can observe, for example, from the file size [11].

4.2.1 Libraries for reading PDF

Given the limited selection of programming languages in which this task can be implemented, we will first look at Python libraries. Python not only has a wide selection of libraries for converting PDF to TXT format but is also interactive, making debugging easier [12].

¹ <https://cw.fel.cvut.cz/b222/courses/b0b36dbs/tutorials/start>

For further processing, it is necessary to be able to:

- Read text flawlessly - it cannot read only half of a word or conversely, add extra characters
- Be capable of reading a subset symbol - which will be used for foreign key verification
- Correctly recognize underlined text - essential for key identification

In the next section, we will test different libraries on various PDF inputs to determine if they meet all our requirements and how the input from each library looks like. We will also try PDFs generated in different PDF generators and operating systems to ensure that these libraries read all PDFs consistently regardless of where they were generated. For demonstration, we will take a part of a file and convert it to TXT format using each library so we can compare them.

Relační model v textové podobě

```
Osoba(jméno, příjmení, PSČ, ulice, město)
Telefon(jméno, příjmení, telefon)
Fk: (jméno, příjmení) ⊆ Osoba(jméno, příjmení)
E-mail(jméno, příjmení, e-mail)
Fk: (jméno, příjmení) ⊆ Osoba(jméno, příjmení)
```

Figure 4.1. Used PDF for analyzing output from libraries

4.2.2 PyPDF2

First library that can be possibly used is PyPDF2². PyPDF2 is a free open-source Python library and in addition to extracting text from a pdf file and is capable of splitting, merging, cropping and transforming the pages of PDF files. Example of conversion of 4.1 with PyPDF2:

```
Relační model v textové podobě Osoba(jméno, příjmení, PSČ, ulice,
město)
Telefon(jméno, příjmení, telefon)
Fk: (jméno, příjmení) Osoba(jméno, příjmení)
```

Figure 4.2. Example of output with PyPDF2 library

4.2.3 Pdftminer

PDFMiner³ is another free open-source Python library and it meant to be primarily tool for extracting information from PDF documents. It has many more options, such as reading formatted text, not just plain text, so it has a much greater potential. Unfortunately, the library itself doesn't have any implemented function for reading underlined text, so with this library, it will be necessary to implement own underlined text reading.

² <https://pypdf2.readthedocs.io/en/3.0.0/>

³ <https://pdftminersix.readthedocs.io/en/latest/>


```

Relační model v textové podobě

Osoba(jméno, příjmení, PSČ, ulice, město)

Telefon(jméno, příjmení, telefon)

Fk: (jméno, příjmení) Osoba(jméno, příjmení)

```

4

4.2.4 PyMuPDF

PyMuPDF⁵ is a Python library for data extention, analysis, conversion, and manipulation of PDF documents. It supports not only PDF but also XPS, OpenXPS, CBZ, CBR, FB2 and EPUB formats. The library promises to be able to find and read various text modifications, but we could not find any directly implemented function for finding underlines.

```

Relační model v textové podobě
Osoba(jméno, příjmení, PSČ, ulice, město)
Telefon(jméno, příjmení, telefon)
Fk: (jméno, příjmení) Osoba(jméno, příjmení)

```

6

4.2.5 PDFAnnots

Pdfannots⁷ is not a library, but an open-source program written in Python that uses the Pdminer library. This program is primarily designed for extracting annotations such as highlights, comments, underlines, etc. from PDF, and was specifically created for scientific articles, on the output. This program seemed to be a very promising candidate for our work at the beginning, but I couldn't replicate the reading of underlined text. It seems that PDFAnnots has underline defined somewhere in it, so it can read only some, whereas we want to be able to read underline everywhere.

4.2.6 Summary

Neither of the libraries proved to be an optimal choice for our purposes. Each of them has at least one drawback, and the most problematic condition is the ability to read underlined text, which, as it turns out, is a more significant issue than it initially appears. At the same time, we found that all the libraries are consistent across the test PDFs. Therefore, the original PDF has no influence on text reading, output formatting, or even on reading subset symbols or underlines.

⁴ The subseq symbol is not visible in thesis, but was visible in terminal

⁵ <https://pymupdf.readthedocs.io/en/latest/tutorial.html>

⁶ The subseq symbol is not visible in thesis, but was visible in terminal

⁷ <https://github.com/0xabu/pdfannots/blob/main/pdfannots/types.py>

Library	Convert to text	Can read subset symbol	Can read underlined text
PyPDF2	✓	×	×
Pdfminer	✓	✓	×
PyMuPDF	✓	✓	×
PDFAnnots	✓	✓	×

Table 4.1. Overview of the advantages of individual libraries

4.3 Reading the underline text in PDF

As mentioned in the previous section, the biggest issue with reading input in PDF format turned out to be underlining. Since PDFAnnots managed to read some underlines, there arose a suspicion that underlining is represented entirely differently and is not part of the text itself.

In the next section, we will try to determine whether underlining is really a part of the text, and if so, how it is represented in the text.

4.3.1 PDF to DOC/X

The first attempt we tried was to convert the PDF to DOCX using Acrobat⁸. A DOCX file is a document format created by Microsoft and contains various textual modifications, including images, tables, etc. [13]. However, as it turned out, the converted file to DOCX retains the underline information because when opened in Microsoft Word⁹, the underlining was still present.

4.3.2 PDF to HTML

Our next attempt was the web format HyperText Markup Language (HTML). According to the HTML specification, [14] *HTML is the World Wide Web's core markup language. Originally, HTML was primarily designed as a language for semantically describing scientific documents.*

First, we tried to convert the PDF file to HTML using an online converter¹⁰. After opening the file, it contained all the underlines, but we couldn't find the `<u>` and `</u>` tags, which represent underlining in HTML files. This led us to suspect that underlining is not part of the text. So, we deleted all the text and its elements and as a result, we were left with a file that, when displayed, only contained underlining. Since the file, apart from the header, only contained images, the underlining in the PDF file must be represented as an image, making it difficult for us to recognize and read it.

Here is the original file for comparison and the file after deleting the text including elements.

Relační model v textové podobě

Osoba(jméno, příjmení, PSČ, ulice, město) _____
 Telefon(jméno, příjmení, telefon) _____
 Fk: (jméno, příjmení) ⊆ Osoba(jméno, příjmení)
 E-mail(jméno, příjmení, e-mail) _____
 Fk: (jméno, příjmení) ⊆ Osoba(jméno, příjmení)

Figure 4.3. Comparison of PDF files before and after text and text elements deletion.

⁸ <https://www.adobe.com/acrobat/pdf-reader.html>

⁹ <https://www.microsoft.com/en-us/microsoft-365/word>

¹⁰ <https://www.pdf.to/html/>

4.4 Suggestion for changing the assignment

The main reason for submitting this assignment in PDF format is its widespread use and the ability to generate it practically anywhere.

As we found out, we cannot guarantee that we will be able to read underlines from it without errors because the representation varies with each PDF generator.

For better input reading, it would be ideal if we could use a format where underlining is directly specified regardless of where it is generated. Additionally, we need the format to represent the subset symbol in some way so that we can read it.

In any case, it proved to be a suitable solution for our purposes because it inherently embeds underlining and is also capable of preserving the subset character. Moreover, HTML is even more widespread than PDF. [15].

Unfortunately, the change in the assignment will also mean one significant disadvantage for us: we will have to wait for the first submissions from students to start testing because last year's assignments will be unusable for us.

4.5 The input from the Conceptual Model task

Because we want to check not only correctness but also consistency between the first and second parts of the semester project, we need the Conceptual Model as input as well. In this regard, our work is facilitated because the program for evaluating the first part of the semester project is already implemented[16], and we can use its output for our purposes. This output also ensures that we will not receive random data but will have consistently structured input each time.

The output from the evaluation program of the first part is divided into six sections, namely Legend, Entity, Relationship, Errors with Table, Detailed Error Analysis, and Relationship Schema, of which we are interested in only the first three sections.

The mentioned legend facilitates parsing of individual entities, their attributes including keys, and subsequently relationships, showing how each item is represented.

```
===== LEGEND =====
[entity] (attributes) &lt;relationship&gt;
{cardinality} |composite identifier|
* - marks a simple key attribute
$ - marks a weak entity
# - marks relationship member of composite key
```

Figure 4.4. Legend of CP-1 output

The next section with entities starts with a line composed of the equal symbol and the word **Entities**. This means that when we want to read entities, we will need to look for this line, and we will start reading entities after it.

```
===== Entities =====
```

Figure 4.5. Entity line

The names of individual entities are in square brackets, corresponding to the legend labeling. After the line with the entity, there is a line with ancestor, if the entity has any, followed by attributes and composite keys. Unlike ancestor or attributes, composite keys have each attribute listed on the next line. These attributes are again labeled according to the legend in vertical bars.

Example n.1:

```
[Zákazník]
  Ancestor: [Uživatel]{complete, overlapping}
  Attributes: ({0..N}sledující)
```

Example n.2:

```
[$Členský příspěvek]
  Attributes: (od, do, výše, rok)
  Composite keys:
    |#zaplatil, rok|
```

The section with entities ends with a line indicating the section with relationships. Our separator this time will be a line with equal symbols and the word Relationships.

```
===== Relationships =====
```

Figure 4.6. Relationship line

Each relationship name is enclosed by `<` and `>`, which in HTML corresponds to the less-than sign and greater-than sign. On the following lines, there are first the frequencies for the given relationship, followed by the name of the entity entering the relationship. If the relationship has any attributes, they are listed on the last line, just like the attributes in entities are listed.

```
&lt;založil&gt;
  {1..1} - [Zákazník] - Description: []
  {0..N} - [Zaměstnanec] - Description: []
  Attributes: (dne)
```

Our reading of relationships will end with a line containing percentages because that marks the separation for the next part with the table of errors, thus concluding the entire reading from this input.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Chapter 5

Project requirements

After reviewing the analyses of our project in the previous chapters, in this section, we will look at the requirements we have for our program. We will recall our goal, examine the process our program will follow, and finally clarify the language requirements.

The goal of the entire program is the automatic evaluation of the second part of the semester project in the Database Systems course. The program should assess the correctness, check if the submitted work makes sense, and ensure that it complies with the given format. Subsequently, based on the output from the first part of the semester project and the input provided by the student, it will evaluate the correctness of the transformation and give the points according to the requirements of the teacher. This part is submitted in HTML format and expects the output in txt format.

5.1 Requirements

In this section, we will describe all the requirements imposed on the program, whether from a user or system perspective. These requirements will help us summarize everything we expect from the program, and based on that, we can propose a solution/design.

- *Input Format* - The program will be able to read the html output from the first task and from the student.
- *Ability to process input* - The program should be able to process input texts
- *Check the transformation* - The program should be able to check the transformation from the ER diagram to the relational model
- *The ability to distinguish incorrect file* - The program should recognize that an inappropriate file had been uploaded as input
- *Recognize incorrect content* - The program should recognize that the content is not a relational model.
- *Ability to recognize extra text* - The program should be able to recognize when it is text for evaluation and when the text is informative or redundant
- *Setting the Evaluation* - The program allows the user to set the point evaluation according to their own judgment.
- *Output Format* - The output of the program will be a .txt file.
- *Output Content* - The program will list all mistakes, including missing parts of the task to given name file.
- *System Support* - The program can be run in BRUTE mode, meaning it will be written in one of the supported languages: C, C++, Python, or Java.

5.2 Design

In this section, we will look at the design of the problem-solving process starting with input files, proceed with initial processing, correction, and finally output files.

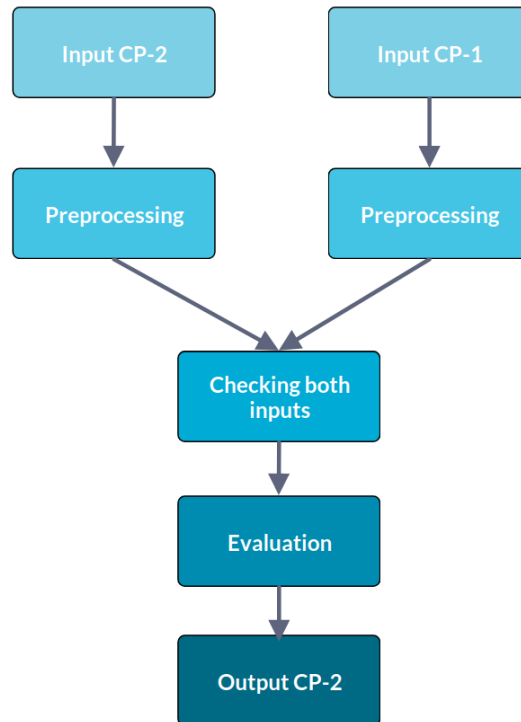


Figure 5.1. Diagram of the task evaluation

1. First, our program takes and preprocesses both inputs, removing unnecessary parts and keeping only the relevant sections needed for checking the correctness of the conversion.
2. In the next step, the program takes both inputs and sequentially checks whether each part of the ER diagram has been correctly converted to the Relational model.
3. Based on this check, the program evaluates and assigns the number of points the student will receive for the assignment.
4. Lastly, the program writes all detected errors into a file according to the specifications, so that instructors can review them, and finally, it outputs the proposed number of points.

5.3 Technical design

As part of the design, we also include a technical overview that summarizes information such as the programming language and environment in which the project will be implemented.

Since the program is intended to be used in the BRUTE system, we are limited by the choice of languages and versions. At the time of designing, it was possible

to write the project in Java version '16.0.2' with the compiler version 'javac 16.0.2', in Python version '3.9.2', or in C/C++ with the compiler 'Debian clang version 11.0.1-2'.

Out of these options, we chose to implement the program in Python. The initial reason for choosing Python was the wide range of libraries available for converting PDF to TXT format. However, this reason became irrelevant after the change in the assignment format submitted by students. Although we could have changed the language, we decided to stick with Python because, at that moment, we did not consider switching to another language.

Chapter 6

Implementation

In this chapter, we will go through the implementation of our project. First, we will cover all the steps of implementation from input to processing. Next, we will look at how the program is run and what files it contains. Lastly, we will review the expected project timeline versus the actual project timeline, thus concluding the implementation.

6.1 Steps of implementation

The next step of our work is the actual implementation of the evaluation program, which we will describe in detail in this section. We will also look at the complications we had to deal with and how we had to adjust the program accordingly.

6.1.1 Classes

Given that we need to store individual information about entities, relationships, and tables, we implemented them as three different classes.

In the following section, we will look at what information each class stores.

```
class Entity:
    def __init__(self, name, print_name, weak):
        self.name = name
        self.print_name = print_name #original name
        self.weak = weak #1 if weak, 0 if not
        self.weak_rel = [] #if weak, what relationship is connected to
        self.ancestor = () #tuple of (ancestor, ancestor_type)
        self.attributes = [] #array of attributes
        self.multival_attributes = [] #array of tuples \
                                     #(name, cardinality)
        self.comp_attributes = [] #array of tuples (name, attributes)
        self.keys = [] #primary key
        self.comp_keys = [] #composite keys
```

```
class Relationship:
    def __init__(self, name, print_name, fst_entity, \
                 fst_relationship, snd_entity, snd_relationship):
        self.name = name
        self.print_name = print_name #original name
        self.fst_entity = fst_entity #first entity in relationship
        self.fst_relationship = fst_relationship #cardinality
        self.snd_entity = snd_entity #second entity in relationship
        self.snd_relationship = snd_relationship #cardinality
        self.attributes = [] #array of attributes
```



```
class Table:
    def __init__(self, name, print_name):
        self.name = name
        self.print_name = print_name #original name
        self.attributes = [] #array of attributes
        self.keys = [] #array of keys
        self.foreign_keys = [] #tuple of (foreing_key,
                                   #origin_attributes, origin_table)
```

All classes also contain methods for adding all non-init attributes to the array. Example for adding keys to Entity we have implemented the method.

```
def new_key(self, key):
    self.keys.append(key)
```

6.1.2 Parsing input CP-1

In this section, we will briefly go through the implementation of parsing and storing relevant data from the CP-1 input. Reading and parsing were quite straightforward because this always has a precisely defined structure that we can rely on being there.

For parsing CP-1 and subsequently CP-2, we used the built-in library 're' ¹.

The parsing of the CP-1 input was implemented iteratively. After opening the file, the program began reading line by line until it reached the relevant part. The program then read individual parts based on specific markers and stored them in classes.

The program stored the names of entities and relationships twice: once in the original form for easier reading and once in a modified form to ensure maximum accuracy during parsing.

Given that there were numerous modifications, we will list all the adjustments and explain why each was made.

- Removal of spaces: Sometimes names consist of multiple words, or a student might accidentally add an extra space.
- Removal of '-' characters: Similar to spaces, a student might use this character if the name of an entity contains multiple words for example 'země-původu'.
- Conversion to lowercase: Standardizing capitalization to ensure uniformity.
- Unicode: To eliminate diacritics, which can complicate validation, we used the Python library Unicode ².

The code for parsing the CP-1 input ends with a line containing percentages, which marks the end of the relevant part of the input. As a return value, it returns all stored entities and relationships.

6.1.3 Parsing problems CP-1 input

The implementation of reading and parsing CP-1 did not go without complications. Over time, we encountered a few problems that caused inconveniences and required us to fix the program.

¹ <https://docs.python.org/3/library/re.html>

² <https://pypi.org/project/Unicode/>

- Composite attribute

According to the legend, the attribute is in regular round brackets, and a composite attribute is nested within additional round brackets, looking like this.

```
Attributes: (vítězný tým, délka(minuty, sekundy), nejlepší hráč)
```

The problem arose when we began testing the program on inputs from students and found that students use round brackets, for example, to indicate types, which should not be in this part of the semester project, or they use them to indicate some format, such as a time format. Example:

```
Attributes: (*název, země původu, datum vzniku (dd.mm.rrrr))
```

We solved this problem by modifying the program for evaluating CP-1 so that a composite attribute is indicated by the characters `/*` and `*/`.

```
Attributes: (vítězný tým, délka/*minuty, sekundy*/, nejlepší hráč)
```

- No attribute

Another issue we did not consider was the possibility that an entity might not contain any attributes. The program for evaluating CP-1 is designed such that if a component, such as an ancestor or a composite attribute, is missing, it will not display it at all. However, it is not the case with attributes. When the table does not contain any attributes, it only outputs a line with the word 'Attributes' and does not even display empty brackets to indicate the absence of any attributes.

```
Attributes:
```

We used parsing with `re.search` function for reading all attributes.

```
attributes = re.search(r"\(.*\)", line).group(0)[1:-1]
                #parsing the attributes
```

Considering that this function returns 'None' when it finds nothing, we added a condition that first checks if it's not 'None' before proceeding with parsing.

- Parts containing newlines

Another significant issue was discovering that some parts contain newlines, causing these parts to be split into multiple lines. Example:

```
Attributes: (*cislo
pasu, {1..N}krestni
jmeno, prijmeni, datum
narozeni)
```

This error was quite challenging to fix and required a complete rewrite of the CP-1 input parsing.

6.1.4 Parsing the CP-2 input

In this section, we will go through the implementation of parsing CP-2 inputs, where students submit their relational models.

Unlike CP-1, in this case, the program is designed to handle various versions of student inputs, as the assignment requirements changed while students were still submitting their work.

- First version

The first version that the program can read is based on our initial expectations of how students would submit their work. We expected students to work in consistent editors and export their work in HTML format instead of PDF. This version is implemented such that, upon opening the file, the program tries to locate the body tags, since all these editors generate a complete HTML file including the header and body. The program then iteratively reads individual lines, stripping out unnecessary tags that are not of interest. Relevant parts, such as tables and foreign keys, are gradually stored in an array of tables, with foreign keys being saved to the respective tables they belong to.

- Second version

The second version that our program can handle follows the example that was later added to the assignment instructions. If the program detects that the student used this example format upon opening the file, it reads the entire file and removes unnecessary tags and text, except for underline tags `<u>`, `</u>` and list item tags ``, `` which always delimit a line.

```
<ul>
  <li>Operace(<u>čas, zakaznik</u>, <u>čas, zamestnanec</u>, <u>čas,
  exemplar</u>, typ_operace) <ul>
    <li>FK: zakaznik Zakaznik(jmeno)</li>
    <li>...</li>
  </ul></li>
  <li>R(...)</li>
</ul>
```

Figure 6.1. Example in assignment

According to the `` and `` tags the program parses individual lines to determine whether they pertain to a table or a foreign key. If the line indicates a table, it creates a new table; if it indicates a foreign key, the key is stored in the last-read table, as it should belong to that table.

6.1.5 Problems with Parsing the CP-2 Input

- Encoding

The first issue we encountered while attempting to read and parse student inputs was different encodings. Until then, we had not considered the possibility that someone might use an encoding other than UTF-8, but in a few cases, windows-1250 encoding appeared.

We resolved this issue by attempting to read the file in UTF-8, and if that failed, we tried reading the file in windows-1250 encoding. Unfortunately, we cannot ensure reading in any arbitrary encoding, as this would require sequentially attempting to open the file in all possible encodings.

- Various Structures

The biggest problem we faced was handling different structures. Despite our efforts to ensure the most reliable and universal reading and parsing, we cannot guarantee that our program will correctly parse any HTML file that is not written according to the examples in the assignment or not generated by an editor. In such cases, there is no certainty that the file will contain a header and body, or that the entire code will not be on a single line, making it impossible to parse line by line.

6.1.6 Checking

After implementing the parsing, we moved on to implementing the correctness check for the conversion from the ER diagram to the relational model.

To facilitate easier point-based evaluation, the program has a separate section for checking tables converted directly from entities and tables converted from relationships.

The program takes all entity names and attempts to find them among the tables, gradually checking if the student correctly converted their attributes and keys, including foreign keys. For foreign keys, it checks if the table actually contains the key and then verifies that it exists in the original table from which the foreign key is derived, ensuring that the student is not referencing a non-existent table or attribute. In the case of checking weak entity types, multivalued attributes, and tables having an ancestor, the program checks these items separately to ensure accuracy.

Next, the program proceeds to check the relationships. If it is not necessary to create a new table for the relationship, it verifies that the student has correctly converted the relationship, which in some cases means adding a foreign key to an existing table. If a new table is required for the relationship, the program first tries to find the table by name and if unsuccessful, searches for the table based on foreign keys, as there is precisely one relationship between two tables.

During the checking process, the program records all students' mistakes, but unfortunately, these mistakes are logged chronologically as the program encounters them.

6.1.7 Problem with Checking

Since we thoroughly described how the individual parts would be converted before writing the checking code, we did not encounter any major complications, just some minor errors that we corrected immediately during implementation. The only problem we had was incorrect reading when a student submitted the assignment in the incorrect format that our program could not read.

6.1.8 Final Points

An important part of our work is to grade the student's submitted work, although this grading will be primarily informative for the teachers.

The point calculation is handled as a dictionary, where the key of each part corresponds to the number in the order they are in the configuration file, and the value is a tuple where the first in the pair is the number of correct elements and the second in the pair is the total number of elements.

The final number of points is then calculated using the formula:

$$\text{points} / \text{number of elements} * \text{number of correct elements}$$

6.1.9 Printing the output

Output printing is divided into several sections, each separated by lines of equal signs.

In the first section, we print the score table, and below the table, the number of correct elements out of the total number of elements is shown to clarify how the score was calculated.

In the second section, all the mistakes that the program detected are listed. These mistakes should be checked by the instructor to determine if they are indeed valid mistakes or if, for example, the student renamed a table, causing the program to incorrectly assess the conversion.

6.1.10 Configuration

Given that the program will be available to all teachers and each evaluates the each parts somewhat differently, we have created the file `config.py`, in which instructors can adjust the number of points for individual parts.

The evaluation of the work is divided into nine parts as follows:

- Table: Tables that are converted from entities
- Keys: Keys in tables converted from entities
- Attributes: Attributes in the tables
- Foreign keys: Foreign keys from tables, including a check to ensure that the attributes in the foreign key match the attributes inside the tables
- Relationship table: Tables that are created from relationships
- Relationship keys: Keys in the tables that are created from relationships
- Relationship foreign keys: Foreign keys in tables that are created from relationships
- Weak entity: Weak entity type
- ISA hierarchy: Tables that have some parent entity

6.1.11 Program parameters

```
usage: main [-h] [-r CP1_INFILE] [-i STUDENT_INFILE]
          [-o OUTFILE]

Parsing the student homework and checking its correct

options:
  -h, --help            show this help message and exit
  -r CP1_INFILE, --cp-input CP1_INFILE
                        Input file - output from CP1.
  -i STUDENT_INFILE, --input STUDENT_INFILE
                        Input file from student, file to be evaluated
  -o OUTFILE, --output OUTFILE
                        Output file (default is output_cp2.html).
```

6.2 Structure of the program

In this section, we will look at the structure of the program, detailing all the files it contains and how they are interconnected.

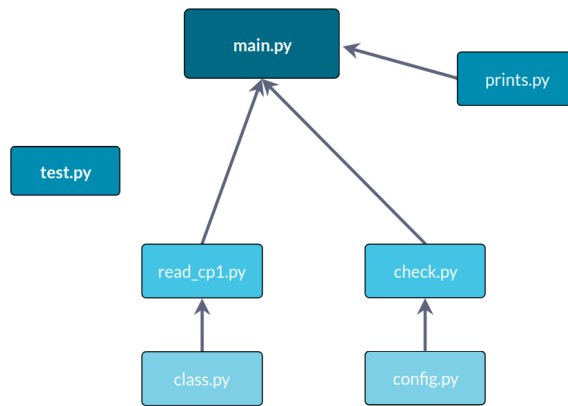


Figure 6.2. Structure of the program

- `main.py` - main file
- `read_cp1.py` - file for reading CP-1 input
- `classes.py` - file containing the Entity, Relationship and Table classes
- `check.py` - file including all checking functions
- `prints.py` - file including all printing functions
- `config.py` - configuration file
- `test.py` - file with tests

6.3 Timeline of the project

Things do not always go according to plan, so in this section, we will examine the planned timeline of the project compared to the actual timeline.

- Expected timeline

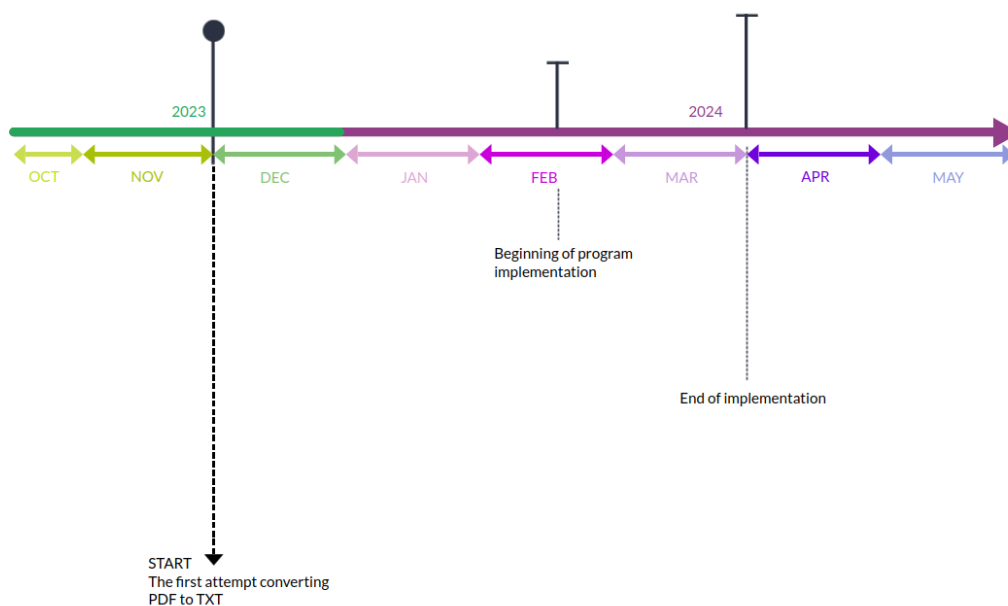


Figure 6.3. Expected timeline

- December 2023 - We expected that after analyzing the problem, we could start looking into converting PDF to TXT format in early December and possibly attempt to read some student submissions.
- February 2024 - With the start of the new semester in the second half of February, we anticipated beginning the implementation of the entire project.
- April 2024 - At the beginning of April, with the deadline for submissions approaching, we expected the entire program to be completed.

■ Timeline of project

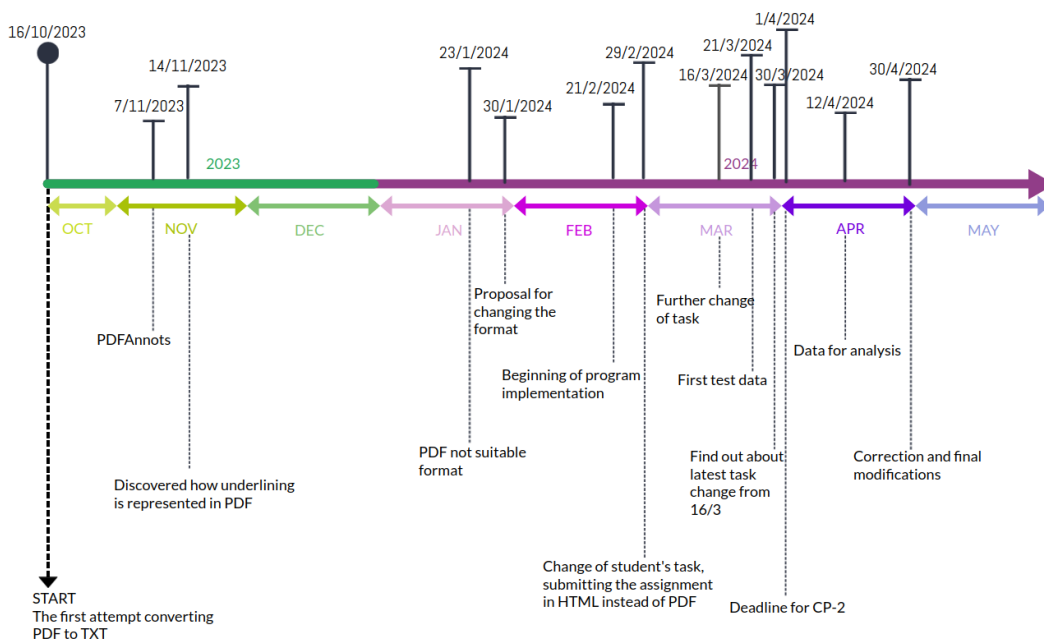


Figure 6.4. Timeline of project

- 16/10/2023 - This was the beginning of the journey into PDF exploration. I searched for and tried several libraries, looked into how to convert PDFs into text, and started investigating how to extract underlined text from PDFs.
- 7/11/2023 - Since none of the libraries proved to be a suitable solution, it was suggested by the supervisor to use the PDFAnnots program.
- 14/11/2023 - After some research on why PDFAnnots reads some underlines and not others, a problem on Stack Overflow was discovered, addressing that underlining in PDFs is handled in various ways, including a black-colored image.
- 23/1/2024 - PDF was not proving to be a suitable format to read from.
- 30/1/2024 - Consideration of using a different format that we could read everything we need from, and one that is ideally well-defined.
- 21/2/2024 - Started implementing the parsing of input from CP-1 and gradually other parts as well.
- 29/2/2024 - Change in the assignment of the semester project. The new requirement is the second part of the semester project to be submitted in

HTML format. With this came the start of implementing parsing of student input, but unfortunately, we don't know what to expect.

- 16/3/2024 - Another change in the assignment without any notice.
- 21/3/2024 - First test data received, the inputs from students looked completely different from what we anticipated, leading to another necessary change in parsing student input.
- 30/3/2024- Discovery of another change in the assignment, further adjustments to parsing student input.
- 1/4/2024 - Deadline for the submission of the second semester project.
- 12/4/2024 - Additional data from two parallel groups for testing and further analysis.
- 30/4/2024 - Returning to the code and making final adjustments, parsing, and printing.

Chapter 7

Testing and results

In this chapter, we will look at the description of the testing and then summarize the errors that we receive from our program after evaluating the assignments.

7.1 Testing

Testing the program was conducted progressively from unit tests to testing larger functions.

The biggest challenge was the regex used for parsing input. It was relatively easy to test on small segments, but for larger inputs, it became quite difficult, and finding unhandled edge cases was not easy. This was because unexpected or unintended input would sometimes be present, and our program would not know how to handle it.

As it later turned out, we initially did not consider that students might use characters in their work that our regex relied on for parsing. This necessitated adjustments to the output in 4.5.

Even after testing and handling everything we identified, we cannot guarantee that the program will not encounter parsing issues with regex in the future. This could be due to further changes in the assignment requirements or new, yet undiscovered errors in student input.

The program will be re-tested before its final deployment in the BRUTE grading system, and further modifications will be made if necessary.

7.2 Results

We received submissions from 42 students across 2 parallel classes for testing our program's functionality and comparing any mistakes we found with those reported by teachers.

We ran the program on each of the 42 inputs, observing the results. However, we encountered issues with 14 students' submissions. Upon closer look, we discovered that some had submitted their tasks in UML format, while others provided files that were formatted in a way that made parsing difficult. These files had a hard-to-define structure, for example were structured in a single line, without tags that could be used for parsing.

The following results are based on 28 student projects on which we were able to run the program.

7.2.1 Types of mistakes

Just as teachers did not identify the origins of certain mistakes during manual grading, our program unfortunately also does not track the source of all mistakes. For instance, it does not check for the presence of a weak entity type, as this error pertains to the correction of the previous part of the semester project. The program also does not distinguish whether a missing attribute in a composite key is marked as an attribute or simply forgotten.

Therefore, we will introduce new designations for the mistakes that we were able to observe from the program. These errors will be labeled in the format Rxx, where xx corresponds to the sequential number.

- R01 - Additional attribute
- R02 - Missing attribute
- R03 - Additional key
- R04 - Missing key
- R05 - Incorrect FK
- R06 - Has one composite key instead of multiple
- R07 - Missing tables
- R08 - Missing relationship
- R09 - Missing FK
- R10 - Incorrectly marked key
- R11 - Wrong FK in SET
- R12 - Incorrect SET
- R13 - Missing table for multivalued attribute
- R14 - Incorrect ISA hierarchy

For an overview of the identified errors, let us look at the graph, which again shows the frequency of these errors among the students.

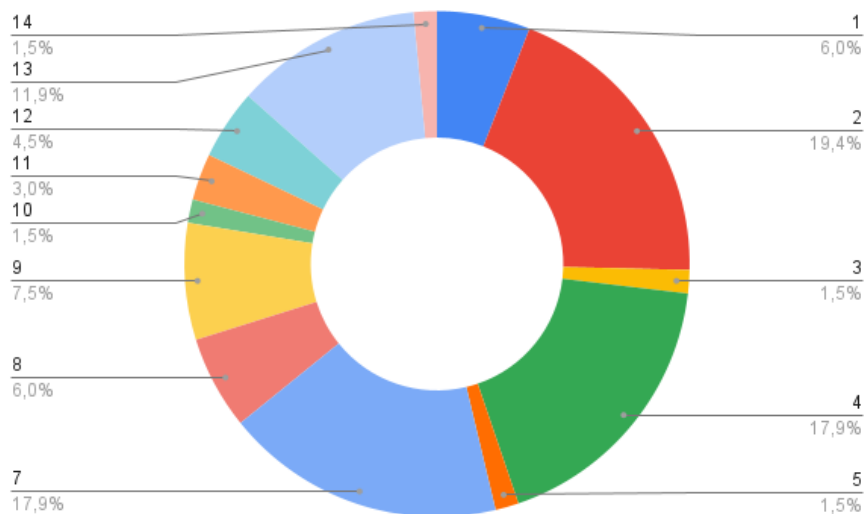


Figure 7.1. Graph with mistakes found by the program.


7.3 Summary

Although we did not expect it, it appears that the most common issue, A15, has practically disappeared. To verify, we briefly reviewed the submitted files, which indeed did not contain this mistakes. This may be due to the addition of example to the assignment or newly created materials for the summer semester 2023/24.

From the graph, it is evident that missing tables were mostly justified by a missing relationship. Another, though less common, reason was a missing multivalued attribute for which a table was not created.

What is surprising is the frequency of missing keys or attributes, for which there are likely more reasons. One of them would be composite attributes or keys, because if a single attribute is missing, the entire composite attribute or key is counted as incorrect.

The remaining errors are represented roughly in the proportion we would expect, but it is quite difficult to compare these data with the data from the instructors. Each person perceives and overlooks the origin of some errors differently, and counts them in various ways. Additionally, there have been so many changes throughout the year that the students' results will be overall influenced by the teaching itself.



Chapter 8

Conclusion

The goal of this work was to attempt to create a program that would evaluate the second semester project in the Database Systems course, which involves converting an ER diagram into a Relational model. However, during the problem analysis, it became apparent that the PDF format is not suitable for submission if the work is to be evaluated by our program.

After changing the assignment requirements, we were able to implement the program with minor complications. Despite our efforts to create reliable parsing, we cannot guarantee in all cases that the program will be able to correctly read and evaluate students' work if the relational model notation is not followed.

Our expectation for the project submissions was that students would write the Relational model in the same document software as the previous assignment, but export their work as HTML instead of PDF. However, it turned out that students approached this task differently, which led to further modifications of the assignment and the addition of examples. Unfortunately, the example contained errors, which brought us to two possible solutions, one of which we implemented.

The first solution we implemented was to create a template that included all parts of an HTML file, including the head and body. The resulting template is available in the attachments and contains a larger example designed to prevent some common mistakes and also provides a template that students can copy and fill out.

The second option to make the task easier for students and to ensure consistency in the input for evaluation is to create a web application where students would not write the text directly but would only enter the names of tables, keys, attributes, and foreign keys. In the application, it would then be possible to generate an HTML file structured the same way as the template we created, and this HTML file could be submitted by the students. This approach might be much more convenient for students, but unfortunately, it requires more time, which we did not have, so we can only propose it as a bachelor's thesis project.



References

- [1] G. Powell. *Beginning Database Desing*. Wiley Publishing, Inc., Indianapolis, Indiana, 2006.
- [2] J. L. Johnson. *Database : models, languages, design*. Oxford University Press, Inc., 1997. ISBN 0-19-510783-7.
- [3] S. Thakur. *E-R Diagrams in DBMS: Components, Symbols, And Notations*. <https://whatisdbms.com/e-r-diagrams-in-dbms-components-symbols-and-notations/>.
- [4] Il-Yeol Song, Mary Evans, and Eui Kyun Park. A Comparative Analysis of Entity-Relationship Diagrams. *Journal of Computer and Software Engineering*. 1995, 3
- [5] P. Stejskal. *Modelovací nástroj pro ER konceptuální návrh databázi*. 2020. https://dspace.cvut.cz/bitstream/handle/10467/87584/F3-BP-2020-Stejskal-Petr-Modelovaci_nastroj_pro_ER_konceptualni_navrh_databazi.pdf?sequence=-1&isAllowed=y.
- [6] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*. 1970, 13 (6), 377–387. DOI 10.1145/362384.362685.
- [7] M. Svoboda. *Database Systems*. 2019. <https://www.ksi.mff.cuni.cz/~svoboda/courses/182-B0B36DBS/>.
- [8] R. Holowczak T. Connolly, C. Begg. *Mistrovství - Databáze: Profesionální průvodce tvorbou efektivních databází*. 2009.
- [9] Deliang Jiang, and Xiaohu Yang. *Converting PDF to HTML approach based on text detection*. In: *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. New York, NY, USA: Association for Computing Machinery, 2009. 982–985. ISBN 9781605587103. <https://doi-org.ezproxy.techlib.cz/10.1145/1655925.1656103>.
- [10] Adobe Systems Incorporated 2008. *Document management — Portable document format — Part 1: PDF 1.7*. First Edition, 2008-7-1. https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/PDF32000_2008.pdf.
- [11] Thomas A. Phelps, and Robert Wilensky. *Two diet plans for fat PDF*. In: *Proceedings of the 2003 ACM Symposium on Document Engineering*. New York, NY, USA: Association for Computing Machinery, 2003. 175–184. ISBN 1581137249. <https://doi-org.ezproxy.techlib.cz/10.1145/958220.958253>.
- [12] A. Radenski. *"Python First": A Lab-Based Digital Introduction to Computer Science*. 2006.
- [13] *.DOCX File Extension*. 2022. <https://fileinfo.com/extension/docx>.

- [14] WHATWG. *HTML: The Living Standard*.
<https://html.spec.whatwg.org/multipage/introduction.html>.
- [15] Dietrich von Seggern, Klaas Posselt, Tamir Hassan, and Thomas Zellmann. *More than just digital paper-hidden features of the PDF format*. In: *Proceedings of the ACM Symposium on Document Engineering 2019*. New York, NY, USA: Association for Computing Machinery, 2019. ISBN 9781450368872.
<https://doi-org.ezproxy.techlib.cz/10.1145/3342558.3351873>.
- [16] A. Škarda. *Automatické vyhodnocování úloh v předmětu Databázové systémy*. 2023.
https://dspace.cvut.cz/bitstream/handle/10467/108683/F3-BP-2023-Skarda-Adam-Automaticke_vyhodnocovani_uloh_v_predmetu_Databazove_systemy.pdf?sequence=-1&isAllowed=y.



Appendix A

Glossary

- CP-1 The first part of semestral work in Database Systems course, the Conceptual Model.
- CP-2 The second part of semestral work in Database Systems course, the Relational Model.
- ER Entity relationship
- PDF Adobe's Portable Document Format

Appendix B

Attachments

B.1 Source code

The zip file contains both the source code and the data for testing.

```
final.zip
+-- read_me.txt
+-- code
| +-- check.py
| +-- classes.py
| +-- config.py
| +-- main.py
| +-- prints.py
| +-- read_cp1.py
| +-- tests.py
+-- data
  +-- stud01
  | +-- cp1_infile.html
  | +-- cp1-krm.png
  | +-- cp1-krm.xml
  | +-- student_infile.html
  +-- stud02
  | +-- cp1_infile.html
  | +-- cp1-krm.png
  | +-- cp1-krm.xml
  | +-- student_infile.html
  +-- stud03
  | +-- cp1_infile.html
  | +-- cp1-krm.png
  | +-- cp1-krm.xml
  | +-- student_infile.html
  +-- stud04
  | +-- cp1_infile.html
  | +-- cp1-krm.png
  | +-- cp1-krm.xml
  | +-- student_infile.html
  +-- stud05
  +-- cp1_infile.html
  +-- cp1-krm.png
  +-- cp1-krm.xml
  +-- student_infile.html
```


Appendix C

Template

Template for submitting assignments for the following years, including an example.

```
<!DOCTYPE html><html>
<head>
<link rel="stylesheet" href="styles.css"></head>
<body>
<!-- EXAMPLE -->
<!-- <ul>
  <li>Osoba(<u>jméno</u>, adresa)</li>
  <li>Zaměstnanec(<u>jméno</u>, adresa, email, mobil)</li>
    <ul>
      <li>FK: (jméno, adresa) Osoba(jméno, adresa)</li>
    </ul>
  <li>Zákazník(<u>jméno</u>, adresa, email)</li>
    <ul>
      <li>FK: (jméno, adresa) Osoba(jméno, adresa)</li>
    </ul>
  <li>Operace(<u>čas, zákazník</u>, <u>čas, zaměstnanec</u>,
    <u>čas, exemplář</u>, typ_operace)</li>
    <ul>
      <li>FK: (zákazník) Zákazník(jméno)</li>
      <li>FK: (zaměstnanec) Zaměstnanec(jméno)</li>
    </ul>
</ul> -->
<ul>
  <li>Table1(<u>key1, key2</u>, attribute)</li>
  <li>Table2(<u>key</u>, attribute)</li>
    <ul>
      <li>FK: (attribute) Table1(attribute)</li>
    </ul>
</ul>
</pre>
</body>
</html>
```

1

¹ The subseq symbol is not visible in thesis, but is present in original file

Example of how the template is displayed in the browser.

- Table1(key1, key2, attribute)
- Table2(key, attribute)
 - FK: (attribute) \subseteq Table1(attribute)