



**MATEMATICKO-FYZIKÁLNÍ  
FAKULTA**  
Univerzita Karlova

**BAKALÁŘSKÁ PRÁCE**

Dennis Pražák

**Návrh a implementace nástroje na vytváření  
diagramů unifikovaných konceptuálních  
schémat multi-modelových a dalších NoSQL  
databázových systémů pomocí prostředků  
schématických kategorií**

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Martin Svoboda, Ph.D.

Studijní program: Informatika

Studijní obor: IPP2

Praha 2023

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... ..

Podpis autora

Děkuji vedoucímu RNDr. Martinu Svobodovi, PhD., za odborné vedení práce a všechny poskytnuté rady a podněty.

Název práce: Návrh a implementace nástroje na vytváření diagramů unifikovaných konceptuálních schémat multi-modelových a dalších NoSQL databázových systémů pomocí prostředků schématických kategorií

Autor: Dennis Pražák

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Martin Svoboda, Ph.D., Katedra softwarového inženýrství

Abstrakt: Tato práce se zabývá vývojem grafické aplikace pro konceptuální modelování databázových schémat bez předem známého paradigmatu. Dosahuje toho pomocí schématických kategorií, které jsou zobecněním dříve známých modelů jako jsou ER a UML. Tyto modely jsou na schématické kategorie převoditelné a aplikace tuto funkcionalitu umožní. Klasický postup tvorby databázového schématu je nejprve návrh konceptuálního modelu např. ER, který je posléze převeden do logické vrstvy např. na tradiční relační model, který je uplatněn v relačních databázích. V dnešní době je k dispozici mnoho nových modelů a prostředky ER často nestačí k jejich popisu. Výsledná aplikace nabídne uživateli možnost modelovat ve známém schématu ER, automatický převod na schématickou kategorii i možnost modelovat přímo schématickou kategorii.

Klíčová slova: konceptuální modelování, schématická kategorie, databázové systémy

Title: Design and implementation of a tool for diagram creation of unified conceptual schemas of multi-model and other NoSQL database systems using schematic categories

Author: Dennis Pražák

Department: Department of Software Engineering

Supervisor: RNDr. Martin Svoboda, Ph.D., Department of Software Engineering

Abstract: This thesis concerns development of a graphical application for conceptual modeling of database schemas without prior knowledge of the target database paradigm. This is achieved by exploiting schema categories, that are a generalization of previously known models such as ER and UML. These models can be converted into schematic categories and the resulting application will enable this functionality. The traditional procedure for creating a database schema is first the design of a conceptual model using e.g. ER, which is then converted into the logical layer, traditionally the relational model, which is in turn applied in relational databases. Many novel database models are available these days and the resources of ER and UML are often not sufficient to describe them. The resulting application will offer the option to model in the familiar ER schema, automatic conversion to a schematic category, and the option to edit or model the schematic category directly.

Keywords: conceptual modeling, schema category, database systems

# Obsah

<b>Úvod</b>	<b>2</b>
<b>1 Existující nástroje</b>	<b>3</b>
1.1 Srovnávací kritéria . . . . .	3
1.2 diagrams.net . . . . .	4
1.3 drawSQL . . . . .	8
1.4 ERDPlus . . . . .	10
1.5 nomnoml . . . . .	11
1.6 Visual Paradigm Online . . . . .	13
1.7 Závěr existujících řešení . . . . .	14
<b>2 Teorie</b>	<b>16</b>
2.1 Entity Relationship . . . . .	16
2.2 Schématická kategorie . . . . .	18
<b>3 Specifikace</b>	<b>20</b>
3.1 Požadavky . . . . .	20
3.1.1 Funkční požadavky . . . . .	20
3.1.2 Nefunkční požadavky . . . . .	22
3.2 Diagram tříd . . . . .	22
<b>Závěr</b>	<b>25</b>
<b>Seznam použité literatury</b>	<b>26</b>
<b>Seznam obrázků</b>	<b>28</b>
<b>Seznam tabulek</b>	<b>29</b>

# Úvod

Úvod

# 1. Existující nástroje

V této kapitole zanalyzujeme některé existující nástroje pro tvorbu diagramů a porovnáme je dle navržených kritérií. Všechny analyzované nástroje jsou dostupné online ve webovém prohlížeči, jelikož v praktické části bude implementován nástroj pro stejné prostředí. Nástroji, které budeme porovnávat, jsou

- diagrams.net [1] vhodné pro tvorbu libovolných diagramů,
- drawSQL [2] určené pro tvorbu relačních schémat,
- ERDPlus [3] k vytváření zejména ER diagramů [4].
- nomnoml [5] k tvorbě UML diagramů [6] psaním textu,
- Visual Paradigm Online [7] pro tvorbu libovolných diagramů.

Před představením existujících nástrojů určíme srovnávací kritéria, dle kterých budeme nástroje analyzovat.

## 1.1 Srovnávací kritéria

Prvním kritériem pro porovnání nástrojů je jejich kategorie, která vypovídá o účelu nástroje a cílové skupině zákazníků. Základní kategorie jsou

- konceptuální vrstva – tyto nástroje jsou většinou určené pro tvorbu ER diagramů, případně jiným způsobem modelují vztahy a atributy entit, na které při datovém modelování vymezujeme svůj diskurz,
- logická (též technologická) vrstva – tyto nástroje umožňují tvorbu diagramů s ohledem na typ struktur, v kterých jsou data uchovávána, např. relační databáze,
- kresba libovolných diagramů – nástroje, které nejsou omezeny téměř žádným standardem či konvencí a umožňují kresbu libovolných diagramů,
- kresba omezených diagramů – nástroje, které umožňují kresbu diagramů omezených na existující schémata (ER, UML, ...).

Dalším kritériem je typ úložiště. Nástroje mohou ukládat svá data do paměti prohlížeče (lokálně pro uživatele), na své servery, nebo používat externí úložiště uživatele, například Google Drive<sup>1</sup>. Čím více různých typů úložiště nástroj podporuje, tím lépe, neboť uživatel může flexibilně zvolit jeho účelům vyhovující způsob uchovávání dat. Pro interaktivní spolupráci s týmem je lepší sdílené úložiště a pro lokální práci je vhodnější lokální úložiště.

Interaktivní spolupráce je dalším důležitým kritériem. U velkých projektů je vývoj modelu urychlen, pokud nástroj spolupráci umožňuje.

Dále budeme porovnávat formát, do kterého nástroj diagram ukládá (pokud k uloženému souboru má uživatel přístup). Může se jednat o serializovaný dokument do dobře známého standardního formátu, nebo o vlastní formát, který je často nakonec také založený na nějakém standardu.

Kromě uložení rozdělané práce do vhodného formátu musí nástroj umožnit export do formátu, který uživatelé využijí pro své účely. Formáty pro export lze rozdělit do

---

<sup>1</sup><https://www.google.com/drive/>

několika kategorií:

- serializovaný formát – většinou se jedná o vlastní formát aplikace a takový soubor nelze jinou aplikací otevřít, ale lze jej programově zpracovat,
- rastrové formáty, např. PNG<sup>2</sup> – mají nejširší využití a podporu, lze je použít v dokumentech a na webových stránkách,
- vektorové formáty, např. SVG [8] – nemají tak rozšířenou podporu, nicméně jsou vhodnější v dokumentech po estetické stránce (zvláště při tištění); dále existují vektorové editory, pomocí nichž lze výsledek libovolně upravovat bez potřeby souboru ve serializovaném formátu; většina webových prohlížečů formát SVG podporuje a soubor vykreslí; do této kategorie lze zařadit i jiné otevřené strukturované formáty, např. VSDX<sup>3</sup>,
- zjednodušený export – některé nástroje šetří práci uživatele tím, že diagram rovnou exportují do HTML<sup>4</sup>, PDF<sup>5</sup> a podobných finálních formátů pro okamžitou aplikaci, přestože uživatel může zvolit jiný formát a finální vytvořit sám,
- schematické formáty, např. SQL<sup>6</sup> – téměř výhradně u nástrojů logické vrstvy; umožňují rovnou vytvářet schémata pro databáze.

Stejně jako u typu úložiště, čím více různých formátů exportu nástroj podporuje, tím lépe, neboť nástroj je flexibilní.

Posledním, neméně důležitým kritériem, je způsob komercializace. Většina volně dostupných nástrojů je nějakým způsobem zpoplatněna, ať už se jedná o jednorázový nebo pravidelný poplatek. Nejčastějším komerčním modelem je verze zdarma s omezenými funkcemi a dále několik placených plánů různé úrovně s odemčenými pokročilými funkcemi. U tohoto modelu je důležité vyrovnat funkce tak, aby byl nástroj použitelný i v bezplatné verzi, a aby byly placené funkce atraktivní pro uživatele. Při srovnávání budeme věnovat pozornost i tomu, jestli jsou placené funkce esenciální.

## 1.2 diagrams.net

Srovnávací kritéria:

- kategorie – kresba libovolných diagramů,
- typ úložiště – lokální, externí, prohlížeč,
- export – serializovaný, rastrový, vektorový, zjednodušený,
- interaktivní spolupráce – částečně podporována (pomocí externích úložišť),
- komercializace – veškeré funkce jsou zdarma a není potřeba uživatelský účet; z jiného pohledu lze počítat cenu externích úložišť, ale ta jsou volitelná.

Nástroj diagrams.net [1], dříve draw.io, je obecný open-source kreslicí nástroj (který však nepřijímá změny od externích vývojářů [9]) vydaný s licencí Apache

<sup>2</sup>Portable Network Graphics – <https://www.w3.org/TR/2003/REC-PNG-20031110/>

<sup>3</sup>Microsoft Visio XML formát založený na ISO 29500 – <https://interoperability.blob.core.windows.net/files/MS-VSDX/%5bMS-VSDX%5d.pdf>

<sup>4</sup>HyperText Markup Language – <https://w3.org/TR/2021/SPSD-html52-20210128/>

<sup>5</sup>Portable Document Format, ISO 32000 – <https://iso.org/standard/75839.html>

<sup>6</sup>Structured Query Language – <https://iso.org/standard/63555.html>



License 2.0<sup>7</sup>, dostupný jako webová aplikace<sup>8</sup> nebo jako desktopová aplikace. Desktopová verze aplikace je sestavena stejným způsobem jako webová, pouze je zabalena pomocí platformy Electron [10] do okna Chromium. Je vyvinut v běžných webových technologiích (JavaScript<sup>9</sup>, CSS<sup>10</sup>, HTML).

Diagramy lze uložit do serializovaného XML<sup>11</sup> formátu `.drawio`. V tomto formátu je pro každý diagram XML element `diagram`, ve kterém se nachází data zakódována do Base64<sup>12</sup>. Tato data jsou komprimována pomocí zlib<sup>13</sup> a obsahují další XML dokument (URL-encoded<sup>14</sup>, tj. zakódovaný), tentokrát již serializaci vlastního diagramu [11]. Formát tak není bez dekomprese čitelný člověkem. Výhodou je, že lze uložit více diagramů do jednoho souboru a každý pojmenovat. Rozhraní k tomu určené je identické s listy souboru tabulkových procesorů, jako Microsoft Excel<sup>15</sup> a Google Sheets<sup>16</sup>.

Jednosouborový program v jazyce Python (viz kód 1.1) přijímá na standardním vstupu base64 řetězec formátu `mxfile` a na standardní výstup vypíše výslednou dekodovanou XML serializaci. Použití algoritmu Inflate na syrová data je vynuceno podle dokumentace zlib<sup>17</sup>. Jedná se o ukázkou postupu dekodování a toho, že formát je otevřený. Organizace JGraph dokonce poskytuje online nástroj<sup>18</sup>, pomocí kterého lze dosáhnout téhož výsledku.

Soubor s diagramy lze také uložit do formátu SVG, který je navíc otevřený a podporují ho jiné nástroje. Uživatel má při exportu k dispozici možnost *Include a copy of my diagram*, která do SVG souboru zahrne již zmíněný Base64 řetězec, ve kterém je diagram serializovaný. Ve výsledku to znamená, že takto exportované SVG soubory umí `diagrams.net` i otevřít a práce na nich může plnohodnotně pokračovat. Toto řešení se nám líbí, protože se jedná o schování vlastního formátu do SVG, který je nejvhodnějším pro přechovávání a zobrazování diagramů.

```
1 from sys import stdin
2 from base64 import b64decode
3 from zlib import decompress, MAX_WBITS
4 from urllib.parse import unquote
5
6 base64_data = stdin.read()
7 deflated_data = b64decode(base64_data, validate=False)
8 # wbits must be -MAX_WBITS which makes zlib use the raw Inflate algorithm
   without header detection
9 urlencoded_inflated_data = decompress(deflated_data, -MAX_WBITS)
10 urldecoded_data = unquote(urlencoded_inflated_data)
11 print(urldecoded_data)
```

Kód 1.1: Dekódování `mxfile`

Dalšími možnostmi exportu a ukládání jsou

---

<sup>7</sup><https://www.apache.org/licenses/LICENSE-2.0>

<sup>8</sup>na adrese <https://app.diagrams.net>

<sup>9</sup>Standardizován jako ECMAScript, ISO 16262 – <https://iso.org/standard/55755.html>

<sup>10</sup>Cascading Style Sheets – <https://www.w3.org/TR/css>

<sup>11</sup>Extensible Markup Language – <https://www.w3.org/TR/xml/>

<sup>12</sup>RFC 2045 §6.8 – <https://datatracker.ietf.org/doc/html/rfc2045#section-6.8>

<sup>13</sup><https://zlib.net>

<sup>14</sup>RFC 3986 §2.1 – <https://datatracker.ietf.org/doc/html/rfc3986#section-2.1>

<sup>15</sup><https://aka.ms/excel>

<sup>16</sup><https://sheets.google.com>

<sup>17</sup><https://www.zlib.net/manual.html>

<sup>18</sup>k dispozici na adrese <https://jgraph.github.io/drawio-tools/tools/convert.html>


- rastrové soubory PNG, JPEG<sup>19</sup>,
- soubor PDF, do kterého je ve vektorovém formátu diagram vložen,
- soubor HTML, do kterého lze podobně jako v SVG data diagramu uložit v serializované formě, případně pouze vložit veřejný odkaz URL na diagram (pokud je použito odpovídající úložiště); v tomto souboru je pak zahrnut JavaScript od diagrams.net, který diagram vykreslí,
- otevřený formát VSDX, původně vyvinutý pro Microsoft Visio.


Zajímavou vlastností exportu do rastrového formátu PNG je, že po otevření v programu diagrams.net je diagram plnohodnotně upravovatelný. Je toho dosaženo tím, že v tEXt chunku<sup>20</sup> je pod klíčovým slovem mxf ile zahrnuta plnohodnotná serializace diagramu.

Ze stejných souborů lze diagramy také importovat, ovšem editovat je lze jen pokud je v nich zahrnut formát drawio, čehož je dosaženo u některých formátů popsaných výše.

Jako úložiště si lze vybrat Google Drive, OneDrive<sup>21</sup>, Dropbox<sup>22</sup>, GitHub<sup>23</sup>, GitLab<sup>24</sup>, paměť prohlížeče a místní úložiště (disk uživatele). Soubor lze ze stejných úložišť i otevřít a importovat, navíc k tomu i z libovolné dostupné URL.

Interaktivní spolupráce je umožněna pouze pokud soubor jako úložiště využívá takové, ke kterému mají přístup zápisu (popř. pouze čtení) všichni účastníci se uživatelé (Google Drive, OneDrive, Dropbox, GitHub, GitLab). Tato úložiště je však nutno manuálně vhodně nastavit (přístup ostatním uživatelům). U všech úložišť je rychlost reflektování změn ostatních uživatelů podobná – vcelku pomalá, protože aplikace musí změny aktivně kontrolovat a načítat.

Menu  Publish chybně napovídá, že se jedná o funkci interaktivní spolupráce. Ve skutečnosti je uživateli jen zobrazen odkaz na soubor ve vybraném úložišti (ale pouze pro Google Drive a OneDrive, jinak je tato možnost vypnuta). Spolupracující uživatel tak musí tento soubor v daném úložišti uložit k sobě (sdíleně), aby mohla spolupráce začít.

Jako další možnost jsme zvažovali desktopovou aplikaci s načteným souborem, který je libovolným externím nástrojem sdílen mezi uživateli. Bohužel, soubor se nepřenačítá automaticky, ale musí být manuálně synchronizován tlačítkem  Synchronize (Alt+Shift+S), které je dostupné pouze v desktopové verzi aplikace. Uživatel je při externí změně souboru upozorněn (avšak ne spolehlivě vždy) červeným nápisem. Algoritmus synchronizace funguje správně a tak, jak uživatel očekává.

Nejlepší způsob dosažení interaktivní spolupráce je dle našeho názoru volba systému pro správu Git<sup>25</sup> repozitářů (GitLab nebo GitHub), protože

1. tato úložiště jsou dostupná jak z webové, tak z desktopové verze aplikace,
2. synchronizace probíhá pomocí systému Git,

<sup>19</sup>Joint Photographic Experts Group, ISO 19566 – <https://iso.org/standard/65348.html>

<sup>20</sup>dle PNG formátu sekce 11.3.4.3 <https://www.w3.org/TR/2003/REC-PNG-20031110/>

<sup>21</sup><https://aka.ms/onedrive>

<sup>22</sup><https://dropbox.com>

<sup>23</sup><https://github.com>

<sup>24</sup><https://gitlab.com>

<sup>25</sup>Systém pro správu verzí Git – <https://git-scm.com>

3. díky použití systému Git lze jednoduše spravovat verze a body v historii při vývoji diagramu.

K poslednímu bodu je třeba podotknout, že jiná webová úložiště také podporují správu verzí, avšak není tak rozvinutá, jako správa systémem k tomu určeným – Git. Diagrams.net sám o sobě správu verzí neobsahuje, jen obvyklé „Undo, Redo“ pro aktuálního uživatele. Úpravy ostatních uživatelů nelze vrátet postupně, lze se pouze vrátit za bod synchronizace.

Uživateli jsou v levém postranním panelu k dispozici standardní tvary ER diagramů, UML diagramů [6], flowchart diagramů a další základní tvary pro kresbu diagramů. Tvary lze libovolně kombinovat a spojovat podržením levého tlačítka a tažením myši z a do kotev na krajích objektů. Každý objekt a spojovací čára má vlastnosti, které lze upravovat v pravém postranním panelu. Upravovat lze přímo i vlastnosti formátu SVG.

Uživatelské rozhraní, které je vidět na obrázku 1.1, je velmi podobné kancelářským aplikacím Google. Je tak přívětivé pro nové uživatele, kteří již s aplikacemi Google dříve pracovali.

Menu **Arrange >> Layout** umožňuje celý diagram aranžovat do zvoleného rozložení (Horizontal Flow, Vertical Flow, Horizontal Tree, Vertical Tree, Radial Tree, Organic, Circle, Org Chart, Parallels). Případně lze zvolit **Apply...**, kde lze aplikovat libovolnou transformaci rozložení<sup>26</sup>. Tato funkce však není perfektní, protože po změně rozložení se jednotlivé prvky diagramu překrývají a uživatel je musí přesunout manuálně do vhodné pozice. Přenastavení rozložení však alespoň položí prvky do zvolené obecné pozice.

V menu **Extras >> Mathematical Typesetting** lze povolit vykreslování matematické notace pomocí knihovny MathJax [12]. Pokud pak v nějakém textovém objektu uživatel zadá například  $(x^2)$ , je vykresleno  $x^2$ . Správné vykreslení matematické notace je pak zachováno ve všech zmíněných exportovaných formátech.

Jako výhody určujeme

- univerzálnost a flexibilita – nástroj lze použít pro tvorbu jakýchkoli diagramů,
- množství podporovaných formátů – export pokrývá téměř všechny možné účely,
- cena – všechny funkce jsou zdarma,
- více diagramů v jednom souboru

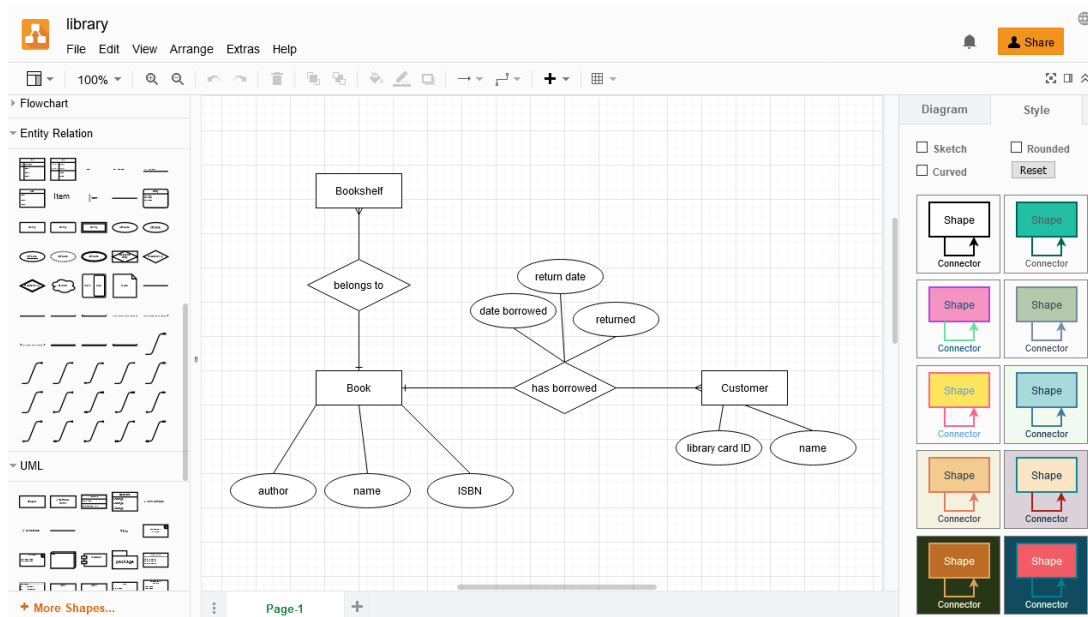
a nevýhodami jsou

- chybějící možnost pro export do (jednoduše) strojově zpracovatelného formátu, nelze tak bez lidské práce diagram převést do logické vrstvy (to je zapříčiněno obecností nástroje, jeho účelem je kresba, ne abstrakce),
- pomalé zobrazování změn při interaktivní spolupráci, zároveň není zpočátku jasné, jak spolupráce dosáhnout.

Výhodou i nevýhodou může být nutnost použití externího úložiště. Pro velké společnosti se může jednat o bezpečnostní opatření, protože diagrams.net k diagramům nemá přístup. Pro malé týmy se může jednat o nevýhodu, protože je potřeba účet na externím webu, nebo jiný způsob sdílení a správa tohoto úložiště.

---

<sup>26</sup>Dokumentace dostupných transformací rozložení <https://jgraph.github.io/mxgraph/docs/js-api/files/layout/hierarchical/mxHierarchicalLayout-js.html>



Obrázek 1.1: Tvorba ER diagramu v aplikaci diagrams.net

## 1.3 drawSQL

Srovnávací kritéria:

- kategorie – logická vrstva,
- typ úložiště – online, poskytované autory produktu
- export – schematický (obecný SQL i platformě specifické formáty), rastrový PNG, serializovaný (JSON [13], v době psaní práce se chystá)
- interaktivní spolupráce – pouze v placené verzi,
- komercializace – omezená verze navždy zdarma, různé měsíčně placené plány.

Nástroj drawSQL [2] je modelovací nástroj pro tvorbu relačních schémat. Aplikace je dostupná ve webovém prohlížeči<sup>27</sup>. Je vyvinuta ve standardních webových technologiích a používá framework Vue.js. Plán zdarma umožňuje tvorbu veřejně přístupných diagramů, které mohou mít maximálně 15 tabulek (entit). Měsíčně placené plány umožňují vytvářet neveřejné diagramy, více (až neomezeně mnoho) tabulek v diagramu, více uživatelů, kteří mohou na diagramu spolupracovat, a přístup k verzovacím nástrojům. K vyzkoušení i používání nástroje je potřeba uživatelský účet.

Hlavní funkcí drawSQL je export schématu do SQL. Proto si uživatel při vytváření diagramu zvolí cílovou databázi, pro kterou schéma tvoří. Výsledné SQL tak bude mít tvar, se kterým cílová databáze umí pracovat. Podporovanými databázemi jsou MySQL<sup>28</sup>, PostgreSQL<sup>29</sup> a SQL Server<sup>30</sup>.

Rozhraní, které je vidět na obrázku 1.2, obsahuje diagram a postranní panel. V postranním panelu lze vytvářet jednotlivé tabulky, definovat jejich sloupce a vlastnosti

<sup>27</sup>na adrese <https://drawsql.app>

<sup>28</sup><https://mysql.com>

<sup>29</sup><https://postgresql.org>

<sup>30</sup>Microsoft SQL Server – <https://aka.ms/sqlserver>

jednotlivých sloupců – typ sloupce, nullability<sup>31</sup>, zda se jedná o primární klíč, unikátní klíč nebo index. Tyto změny se v reálném čase reflektují v diagramu, ve kterém může uživatel jednotlivé sloupce spojovat, čímž vytváří cizí klíče. Pozici těchto lomených čar lze upravovat pouze posunutím tabulky v diagramu. Pokud je cizích klíčů víc, začne být diagram velmi nepřehledný.

Diagram lze importovat ze souboru SQL stisknutím `File >> Import`. Stisknutím tlačítka `File >> Export` se otevře nabídka Export, ve které může uživatel diagram exportovat do SQL své předem zvolené databáze, nebo do rastrového obrázku ve formátu PNG. Vývojáři aplikace plánují implementovat také export diagramu pomocí serializace do formátu JSON. V nabídce Export je navíc možnost nechat si vygenerovat platformně specifický kód jako například migrační třídy pro Laravel<sup>32</sup>, definice modelů pro Laravel, a migrační schémata pro AdonisJS<sup>33</sup>.

Interaktivní spolupráce je k dispozici pouze v placené verzi. Dle našeho názoru je interaktivní spolupráce hlavní funkcí tohoto nástroje oproti konkurenčním relačním modelovacím nástrojům. Některá integrovaná vývojová prostředí (např. Visual Studio<sup>34</sup>) obsahují nástroj pro relační modelování i generování databázového schématu. Hlavním omezením těchto nástrojů je však absence interaktivní spolupráce, jedná se spíše o spolupráci iterací. Proto považujeme určení interaktivní spolupráce za placenou funkci za negativní rozhodnutí pro využitelnost nástroje v relaci s konkurencí.

Web drawSQL také zveřejňuje šablony modelů<sup>35</sup> (jedná se spíše o příklady). Šablony jsou většinou potenciální modely známých produktů (např. WordPress<sup>36</sup>) a tvoří je autoři drawSQL. Tuto funkci považujeme za výhodu, protože společnosti a individuální vývojáři se mohou inspirovat existujícími a ověřenými řešeními, případně nezačínat se svým modelem od nuly.

Závěrem určíme výhody drawSQL:

- příjemné uživatelské rozhraní (viz obrázek 1.2),
- možnost určení typu relace, o sémantiku se aplikace stará sama (one-to-one, one-to-many, many-to-many),
- několik platformně specifických generátorů modelu,
- šablony a příklady existujících modelů

a nevýhody:

- nelze upravit ani přesunout lomené čáry spojující cizí klíče, což způsobuje chaos pokud je v diagramu větší množství entit,
- interaktivní spolupráce pouze v placeném plánu,
- správa verzí pouze v placeném plánu,
- k vyzkoušení nástroje je potřeba uživatelský účet,
- podporuje pouze relační databáze.

<sup>31</sup>nullability je příznak, který určuje, zda lze sloupec v řádku nastavit na hodnotu NULL

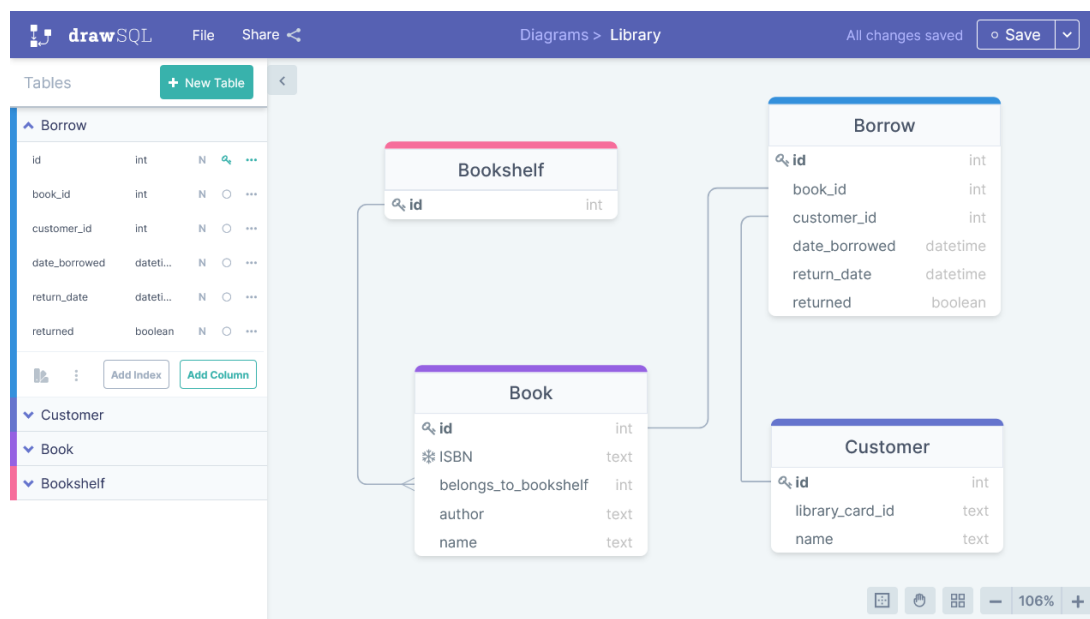
<sup>32</sup>Framework pro PHP – <https://laravel.com>

<sup>33</sup>Framework pro Node.js – <https://adonisjs.com>

<sup>34</sup>Vývojové prostředí Microsoft Visual Studio – <https://visualstudio.microsoft.com>

<sup>35</sup>na adrese <https://drawsql.app/templates>

<sup>36</sup><https://wordpress.com>



Obrázek 1.2: Tvorba diagramu v drawSQL

## 1.4 ERDPlus

- kategorie – logická vrstva,
- typ úložiště – online, poskytované autory produktu
- export – rastrový PNG,
- interaktivní spolupráce – není,
- komercializace – zdarma.

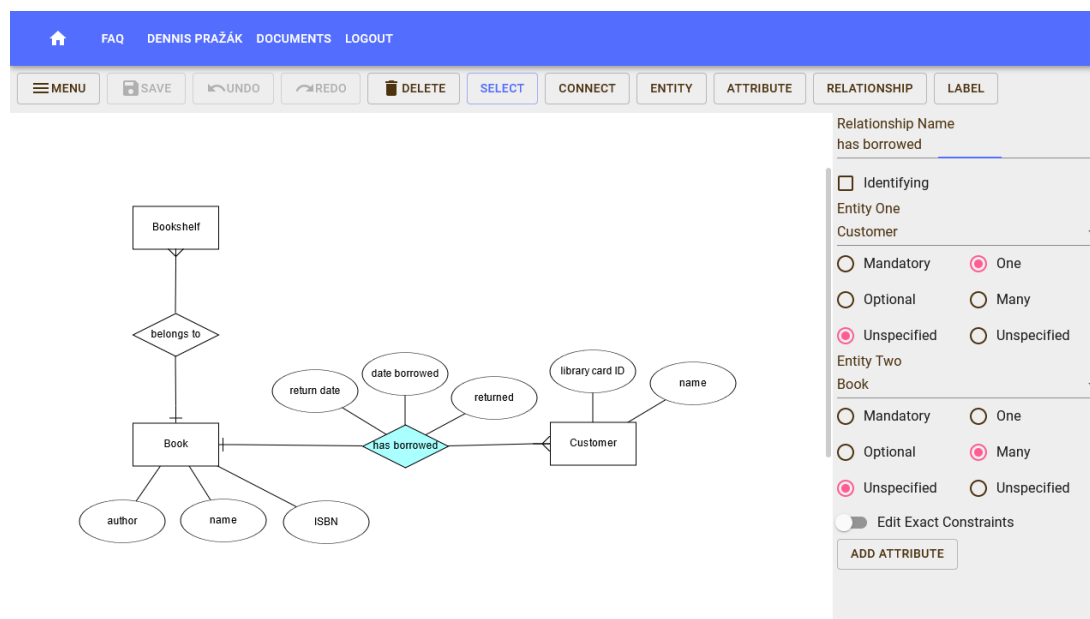
Nástroj ERDPlus [3] je modelovací nástroj pro tvorbu ER diagramů, relačních schém a hvězdicových schém. Aplikace je dostupná ve webovém prohlížeči<sup>37</sup>. Její uživatelské rozhraní je tedy vyvinuto ve standardních webových technologiích – HTML, CSS a JavaScript. Dále využívá framework React [14] pro tvorbu rozhraní v jazyce JavaScript.

ERDPlus lze používat bez založení uživatelského účtu a vytvořený diagram exportovat do speciálního formátu erdplus, nicméně uživatel tak přijde o možnost využití úložiště diagramů na serveru aplikace. Diagramy (ERDPlus je nazývá *dokumenty*) lze organizovat do složek a podsložek. Služby ERDPlus včetně úložiště nejsou žádným způsobem zpoplatněny.

Tvorba ER diagramů je intuitivní s jednoduchým uživatelským rozhraním, které je vidět na obrázku 1.3. Uživatel má na výběr mezi vytvořením entity, atributu, relace, spojení mezi těmito objekty a jednoduchého textového popisku. V pravé části rozhraní se nachází panel s vlastnostmi zvoleného objektu. V tomto panelu může uživatel také rychleji tvořit atributy entit a relací. Při zvolení relace lze v panelu zvolit entity, které mají být v relaci, a spojení je pak automaticky vytvořeno. Zároveň lze zvolit jednotlivé multiplicity relace.

Soubor s diagramem je v úložišti reprezentován vlastním formátem erdplus. Jedná

<sup>37</sup>na adrese <https://erdplus.com>



Obrázek 1.3: Tvorba ER diagramu v ERDplus

se o textový soubor, jehož obsahem je JSON reprezentace diagramu. Diagram lze exportovat do rastrového formátu PNG.

Zajímavou funkcí je také převod do relačního schématu. Tato funkce je dostupná pouze tehdy, když uživatel ER diagram uloží na server ERDPlus. Poté zvolí možnost *Convert to Relational Schema* a ERDPlus vytvoří nové relační schéma. Z relačních schémat lze podobně vygenerovat SQL.

Vlastnoruční tvorba relačních diagramů probíhá podobně. Uživatel může tvořit tabulky, přidávat jim sloupce a v tabulkách volit primární klíče v postranním panelu. Pomocí tlačítka *Connect* lze poté přidat cizí klíč, který odkazuje do jiné tabulky tažením myši. ERDPlus do tabulky přidá všechny primární klíče, které cílová tabulka obsahuje, jako nové sloupce. K vytvoření cizího klíče, který odkazuje na stejnou tabulku (tzv. rekurzivní klíč) slouží tlačítko *Recursive Key* ve vlastnostech tabulky. Hvězdicovým schématům se věnovat nebudeme, protože jsou mimo rozsah této práce.

Výhody:

- převod diagramu z ER do relačního diagramu,
- jednoduchost a intuitivnost procesu kresby diagramu

a nevýhody:

- relační diagram bývá nepřehledný, nelze měnit pořadí jednotlivých definovaných sloupců v tabulce,
- chybí vektorový export,
- diagramy nelze stylizovat.

## 1.5 nomnoml

- kategorie – kresba omezených diagramů – UML,
- typ úložiště – online,

- export – rastrový PNG, vektorový SVG,
- interaktivní spolupráce – není k dispozici,
- komercializace – zdarma s otevřeným zdrojovým kódem.

Nástroj nomnoml je modelovací nástroj pro tvorbu UML diagramů dostupný ve webovém prohlížeči<sup>38</sup>. Jeho klíčová vlastnost je, že místo interakce myši s webovou aplikací probíhá kresba deklarativně – psaním.

Uživatelské rozhraní (viz obrázek 1.4) se skládá z oblasti pro textový vstup, nad kterou je zároveň (v reálném čase) vykreslován výsledný diagram. V pravé horní části se nachází několik tlačítek, při kliknutí na některé z nich se vždy otevře pravý postranní panel s odpovídajícími informacemi a funkcemi. První tlačítko ukazuje rychlý přehled jazyka, ve kterém se má diagram definovat. Druhé tlačítko odhalí kompletní referenci k tomuto jazyku. Dále lze najít tlačítka pro export, sdílení a uložení do místního úložiště uživatele.

Jazyk diagramů je velmi jednoduchý. Skládá se z definic entit a jejich relací. Uživatel může vyjít z úvodního diagramu, který se ukáže při navštívení hlavní stránky nástroje. Pro ukázkou, definice entity vypadá následovně

```
[<abstract> Entita|soukromaSlozka; soukromaSlozka2|verejnyAtri-  
but].
```

Svislá čára odděluje kategorie atributů, může jich být neomezené množství. Entita je definována jako abstraktní, což také ovlivní její výsledný styl vzhledu v diagramu.

Dále se v jazyce definují vztahy mezi entitami [Entita]->[Entita2]. Různé šipky mají různé významy. Vztah -> je *asociace*, dále o-> je *agregace*, apod.

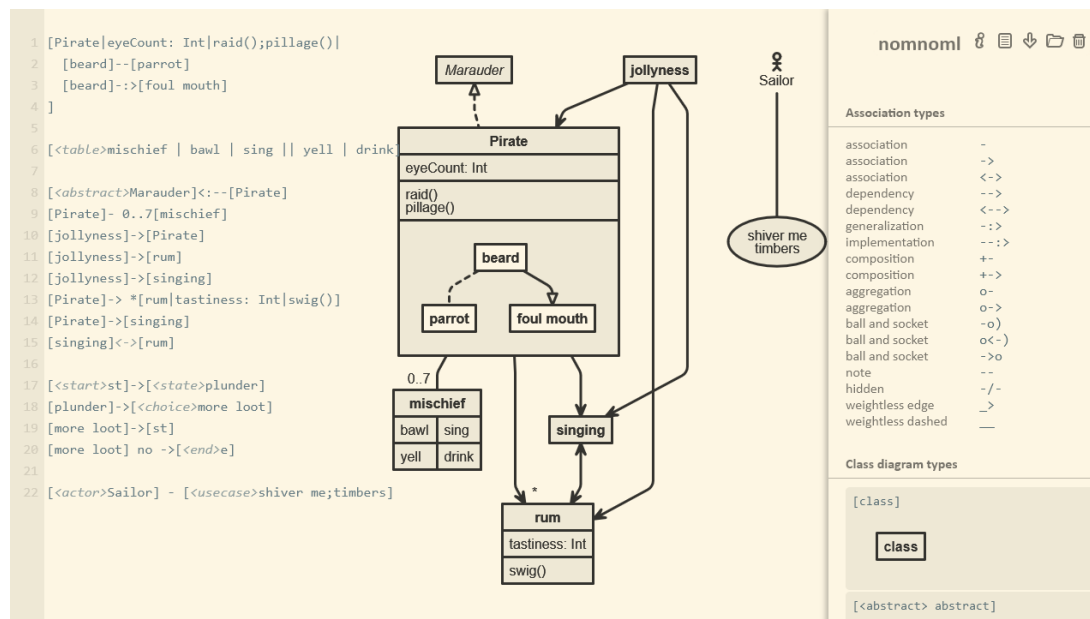
V jazyce lze také deklarovat direktivy, začínající znakem #. Těmi lze upravit vzhled, vytvořit nové styly, a nastavit algoritmy, kterými bude zvoleno rozložení entit v diagramu. Algoritmy lze nastavit direktivou #ranker a na výběr je ze tří možností: network-simplex, tight-tree, longest-path. Nástroj nomnoml používá k vykreslování diagramu knihovnu dagre<sup>39</sup> pro JavaScript, jejíž vývoj byl však ukončen. Z dokumentace této knihovny vyplývá, že možnost *ranker* mění algoritmus, který vrcholům v grafu (diagramu) přiřazuje důležitost, která se pak odráží v pořadí zobrazení entit. Definitivní význam této možnosti není z dokumentace zřejmý, u nástroje nomnoml lze však pozorovat změnu v pořadí a vzdálenostech entit (délce spojovacích čar). Uživatel tak může vyzkoušet různá rozložení a použít to, které je vizuálně nejpřehlednější.

Diagram lze exportovat do formátu PNG a dále podobně jako v sekci 1.2, nomnoml také umožňuje export do SVG se zakomponovaným zdrojovým kódem. Uživatel tak může diagram distribuovat v tomto formátu a zároveň tento formát v nástroji nomnoml i otevřít a plnohodnotně pokračovat v práci. Podobně je možné sdílet odkaz přímo na vytvořený diagram ve službě nomnoml. Odkaz je vytvořen tak, že do URL je jako parametr source zapsán přímo zdrojový kód diagramu. Výsledná adresa URL je tak velice dlouhá, ale služba nomnoml nemusí diagramy ukládat, stačí je vykreslit z dat v odkazu. Přestože se nejedná o opravdové online úložiště, kategorizovali jsme ho tímto způsobem. Rozdělaný diagram lze také uložit do paměti prohlížeče. Díky těmto mechanismům stačí službě nomnoml staticky poskytovat pouze klientskou stranu své aplikace a přesto umožňuje ukládání a sdílení práce.

<sup>38</sup>na adrese <https://nomnoml.com>

<sup>39</sup><https://github.com/dagrejs/dagre>





Obrázek 1.4: Tvorba UML diagramu v nomnoml

Nástroj nomnoml je tedy inovativní svým přístupem ke kresbě diagramů. U složitých diagramů se však nutně ve zdrojovém kódu uživatel ztrácí, protože nomnoml nenabízí žádné dělení či kompozici tohoto kódu. S vizuální reprezentací grafu nelze přímo (např. myší) pracovat, veškeré rozložení a orientaci diagramu tak musí uživatel nechat na algoritmu nástroje. Nástroj dále nenabízí ani možnost pracovat s několika diagramy najednou. Z těchto důvodů určujeme produkt jako vhodný pouze pro jednotlivce a tvorbu méně rozsáhlých UML diagramů.

## 1.6 Visual Paradigm Online

Srovnávací kritéria:

- kategorie – kresba libovolných diagramů,
- typ úložiště – online, lokální, externí, prohlížeč,
- export – serializovaný, rastrový, vektorový,
- interaktivní spolupráce – podporována,
- komercializace – verze zdarma, měsíčně/ročně placené plány několika úrovní.

Nástroj Visual Paradigm Online<sup>40</sup> je proprietární produkt od firmy Visual Paradigm. Tato firma je známá svou stejnojmennou desktopovou aplikací, která má stejný účel. Zde se zaměříme na její online verzi.

Visual Paradigm Online je nástroj pro tvorbu různých typů diagramů. Mimo těch je určen i pro úpravu fotografií, tvorbu koláží, design infografiky, příspěvků na sociální sítě, uživatelských rozhraní, plakátů, dárkových poukazů, apod. Funkcionalita kromě tvorby diagramů je mimo rozsah této práce.

Nástroj poskytuje předlohy tvarů pro diagramy tříd, use case diagramy, sekvenční diagramy, diagramy aktivit, diagramy nasazení (deployment), ER diagramy a velmi

<sup>40</sup><https://online.visual-paradigm.com>

mnoho dalších. Uživatel si před tvorbou diagramu musí vybrat jednu z těchto kategorií, čímž vymezí tvary, které jsou v uživatelském rozhraní dostupné k použití v levém postranním panelu. Ve stejném panelu lze ale posléze přidat libovolnou další kategorii tvarů dle výběru uživatele.

Nástroj je svým rozhráním, designem a chováním nápadně podobný produktu diagrams.net ze sekce 1.2. Stejně tak spojování tvarů čarami má velmi podobné chování. Jeden rozdíl je ten, že kresba spojení z jednoho tvaru do toho samého (tvorba „cyklu“) je velmi nepředvídatelná. Čára se při tom samovolně přemísťuje do všech možných stran, až nakonec zůstane přilepena na jedné ze stran tvaru.

Diagramy lze uložit do formátu vpd, který při otevření v textovém editoru připomíná base64 kód. Vypadá velice podobně jako serializace pro diagrams.net ze sekce 1.2. Nejedná se však o validní base64 kód, protože ten kóduje 3 bajty do 4 znaků, jinak používá výplň 0-2 znaky =. Nicméně vpd obsahuje i sekvence jako je M=Q8, které tomu neodpovídají. Jedná se tak nejspíš o nezdokumentovaný proprietární formát.

Diagramy lze exportovat do formátu SVG a PDF, do kterých lze volitelně uložit i serializaci diagramu ve formátu vpd a případně tak plnohodnotně přenést editovatelný diagram i s jeho vektorovou reprezentací. Rastrový export je možný ve formátu JPEG nebo PNG. Stejně jako u diagrams.net ze sekce 1.2 je do PNG volitelně přidána serializace diagramu v tEXt chunku formátu PNG. Dokonce je použito stejné klíčové slovo identifikující tuto serializaci – mxfile. To vzbuzuje podezření, že se jedná o stejný formát, jako používá diagrams.net ze sekce 1.2, tedy formát serializace modelu diagramu knihovny mxGraph<sup>41</sup>. O tom, že Visual Paradigm Online používá stejný formát jsme ale nenašli žádné zmínky. Dále, jak už bylo zmíněno, vpd nemá stejnou strukturu jako mxfile<sup>42</sup>.

Komerencializace Visual Paradigm Online je rozdělena do několika platebních úrovní. Každá úroveň má určenou cenu za uživatele za měsíc. Úrovně jsou dle ceny a úplnosti funkcí vzestupně Free (zdarma), Starter, Advance, Combo (všechny funkce). Úroveň Free obsahuje i prémiové šablony, ale je v nich zobrazen vodoznak. Ten je v placených plánech odstraněn. Placené plány navíc obsahují více typů diagramů a více tvarů k použití, rastrový export vyššího rozlišení, historii verzí diagramu (při spolupráci) a další.

## 1.7 Závěr existujících řešení

Přehled analýzy zmíněných existujících řešení je vidět v tabulce 1.1.

---

<sup>41</sup><https://jgraph.github.io/mxgraph/>

<sup>42</sup><https://drawio-app.com/extracting-the-xml-from-mxfiles/>

název produktu	diagrams.net	drawSQL	ERDPlus	nomnoml
kategorie (vrstva)	libovolné d.	logická	konceptuální	omezené d.
serializovaný ex.	ano	ne <sup>a</sup>	ano	ano
rastrový ex.	ano	ano	ano	ano
vektorový ex.	ano	ne	ne	ano
schematický ex.	ne	SQL	SQL	ne
zjednodušený ex.	HTML, PDF	ano <sup>c</sup>	ne	ne
poskytuje úložiště	ne <sup>b</sup>	ano	ano	ano <sup>d</sup>
paměť prohlížeče	ano	ne	ne	ano

- a. plánovaná funkce
- b. využívá úložiště třetích stran
- c. platformně-specifický scaffolding – automatické generování kódu pro specifické platformy a programovací jazyky
- d. diagram je uložen v URL, pomocí které lze diagram sdílet

Tabulka 1.1: Srovnání existujících řešení

## 2. Teorie

V této kapitole představíme potřebné teoretické koncepty, kterých bude využívat výsledná aplikace. Mezi tyto koncepty patří Entity Relationship (ER), schematická kategorie včetně teorie kategorií a vizuální diagram schematické kategorie.

### 2.1 Entity Relationship

Datový model Entity Relationship (ER) poprvé představil Chen už v roce 1976 [15]. Od té doby se však ER vyvíjel, jak se potřeby datového modelování rozšiřovaly. ER není standardizováno, ale jednu moderní verzi představili Atzeni, Ceri, Paraboschi a Torlone [16, s. 163-179]. Na jejich ER modelu založíme ten náš, který zde popíšeme.

V tabulce 2.1 jsou vyobrazeny jednotlivé konstrukty ER modelu. Zde blíže popíšeme sémantiku jednotlivých konstruktů:

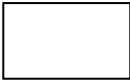
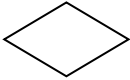

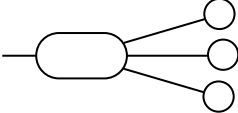

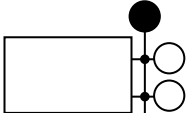
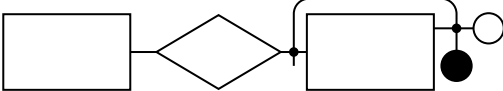
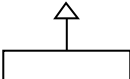
- Entitní typ (Entity Type) reprezentuje entitu. Každá entita má jméno.
- Vztahový typ (Relationship Type) reprezentuje vztah mezi dvěma a více (ne nutně různými) entitami. Každý vztah má jméno.
- Atribut (Attribute) reprezentuje atribut/vlastnost entitních nebo vztahových typů. Každý atribut má jméno.
- Složený atribut (Composite Attribute) je atribut, který má sám atributy. Zakazujeme však další větvení, tedy atributy složeného atributu už samy nemohou být složené. Každý složený atribut má sám jméno, podobně jako jeho vlastní atributy.
- Kardinalita (Cardinality) je dvojice  $(a, b) \in \{0, 1\} \times \{1, *\}$ , kde  $a$  nazýváme minimální kardinalita (spodní hranice) a  $b$  maximální kardinalita (horní hranice). Kardinalitu musí mít všechny atributy a každý účastník vztahu. Výchozí kardinalita je  $(1, 1)$  a ve schématu se většinou neuvádí. Spodní hranice 0 znamená, že účast je volitelná; hranice 1 znamená, že účast je povinná. Horní hranice 1 znamená, že účast je nejvýše jedna; hranice  $*$  znamená, že účastí je libovolný počet.
  - Hranice kardinalit pro jednotlivé účastníky vztahů vyjadřují minimální a resp. maximální počet výskytů jednotlivých instancí účastníků v tomto vztahu.
  - Hranice kardinalit u atributů vyjadřují minimální a resp. maximální počet hodnot atributu, které se vztahují ke každé instanci entity/vztahu.
- Identifikátor (Identifier) umožňuje jednoznačně rozlišit (identifikovat) instance entit. Pro každý entitní typ je povinný alespoň jeden identifikátor, ale může jich být více. Každý identifikátor je tvořen buď
  - jedním nebo více atributy daného entitního typu; takový identifikátor nazýváme interní, nebo
  - jedním, nebo více vztahovými typy, jehož se daná entita účastní, případně kombinací s předchozím; takový identifikátor nazýváme externí.

Entitní typy, které nemají ani jeden interní identifikátor (musí mít tedy externí), nazýváme slabé entitní typy. Pokud mají interní identifikátor, nazýváme je silné entitní typy.

Atributy v ER by měly být pouze elementární vlastnosti. Například pokud by měla mít entita „Zákazník“ atribut „Fyzická adresa“, může být vhodnější fyzickou adresu modelovat jako entitu, neboť má sama atributy jako „Ulice“ a „Město“. Pokud ale víme, že jiná entita nebude v modelu mít fyzickou adresu, můžeme ji případně modelovat jako složený atribut.

U kardinality poznamenáme, že se v ER modelu často dovoluje použít jako hranice libovolná nezáporná celá čísla, tedy  $(a, b) \in \mathbb{N}_0 \times (\mathbb{N}_0 \cup \{*\})$ , tž.  $a \leq b$  (dodefinujeme  $\forall a: a < *$ ). Dají se tak vyjádřit přesnější omezení, např. že jeden uživatel může mít maximálně 5 bankovních účtů. Ve většině případů ale stačí námi definované hranice kardinality, které vyjadřují volitelnost/povinnost pro spodní hranici a jednočetnost/mnohočetnost pro horní hranici. Toto vymezení nám umožní vyjádřit čtyři nejdůležitější případy, nad kterými se při modelování uvažuje.

Dále upozorníme, že místo  $*$  se v ER modelu může použít symbol  $n$  nebo  $N$  pro vyjádření „libovolného počtu“. Důležitá je ale konzistentnost, aby se v jednom modelu nevyskytovaly dva různé symboly, což by mohlo zmást čtenáře. V této práci budeme používat pouze symbol  $*$ .

Konstrukt	Grafická reprezentace
Entitní typ	
Vztahový typ	
Atribut	
Složený atribut	
Interní identifikátor	 
Externí identifikátor	
Zobecnění	

Tabulka 2.1: Grafická reprezentace konstruktů ER modelu, upraveno a přeloženo [16, s. 164]

## 2.2 Schématická kategorie

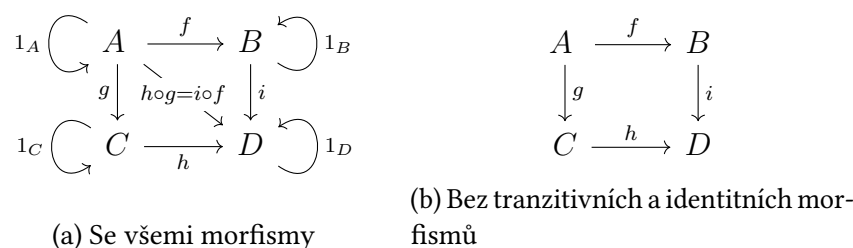
Nejdříve popíšeme kategorii z teorie kategorií, na níž je schématická kategorie založena.

Kategorie je matematická struktura, která zobecňuje ostatní struktury. Umožňuje mimo jiné studovat vztahy mezi nimi. Poprvé byla představena Eilenbergem a MacLanem v roce 1945 [17].

Kategorie  $C = (\mathcal{O}, \mathcal{M}, \circ)$  se skládá z

- třídy objektů  $\mathcal{O}$ ,
- třídy morfismů  $\mathcal{M}$ ; každý morfismus  $f \in \mathcal{M}$  má zdrojový objekt  $A \in \mathcal{O}$ , cílový objekt  $B \in \mathcal{O}$  a říkáme  $f : A \rightarrow B$  ( $f$  je morfismus z  $A$  do  $B$ ),
- operace skládání  $\circ : \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}$ ; pro každé dva morfismy  $f : A \rightarrow B, g : B \rightarrow C$  musí  $g \circ f \in \mathcal{M}$  (tranzitivita); pro tuto operaci navíc platí axiomy:
  - asociativita – pro morfismy  $f : A \rightarrow B, g : B \rightarrow C, h : C \rightarrow D$  platí  $h \circ (g \circ f) = (h \circ g) \circ f$ ,
  - identita – pro každý objekt  $A$  existuje identita  $1_A$ , tž.  $f \circ 1_A = f = 1_B \circ f$  pro každý morfismus  $f : A \rightarrow B$ .

Kategorie je často vizuálně reprezentována schématem připomínajícím orientovaný graf, kde vrcholy jsou objekty a hrany morfismy. Příklad této vizualizace je na obrázku 2.1a. Jedná se o kategorii se čtyřmi objekty  $A, B, C, D$ . V této práci vizualizaci kvůli schématické kategorii budeme zjednodušovat tak, že v ní vynecháme určité morfismy, které budeme považovat za implicitní. Těmito morfismy jsou takové, které jsou v kategorii jen kvůli tranzitivitě, a takové, které jsou v kategorii jen kvůli splnění axiomu identity. Někdy budou tyto morfismy důležité, v takovém případě je v obrázku explicitně uvedeme. Zjednodušená vizualizace je na obrázku 2.1b. Všimněme si, že identity (např.  $1_A$ ) a složené morfismy (např.  $i \circ f$ ) ve schématu nejsou, tedy jsou implicitní.



Obrázek 2.1: Příklad kategorie

Schématická kategorie je zobecnění databázových schémat založená na teorii kategorií. Jedná se o kategorii, jejíž objekty odpovídají jednotlivým entitním typům, atributům a vztahovým typům ER schématu. Morfismy odpovídají vztahům mezi těmito objekty, přičemž aby byly splněny axiomy kategorie, musí se přidat navíc identity a tranzitivní uzávěr těchto morfismů. Tento koncept společně s algoritmem převodu z ER schématu do schématické kategorie uvádí Martin Svoboda, Pavel Čontoš a Irena Holubová [18]. Pro naše účely se v této práci nebudeme řídit přesně podle této originální publikace, nýbrž schématickou kategorií lehce upravíme.

Schematická kategorie se tedy skládá z třídy objektů a třídy morfismů.

Objekt je čtveřice (identita, název, data  $D$ , množina identifikátorů  $I$ ). Identita je libovolný symbol (např. z  $\mathbb{N}$ ), který rozlišuje a unikátně identifikuje objekty, které mají ostatní složky totožné. Název popisuje textovým řetězcem o jaký objekt se jedná a je zde pro uživatele (čtenáře schematické kategorie). Data je množina tzv. *properties* a platí  $I \subseteq \mathcal{P}(D)$ , tedy je to nadmnožina jednotlivých *properties* zmíněných v identifikátorech. Každý identifikátor sestává z *properties* a množina všech identifikátorů je pak množina sestávající z identifikátorů daného objektu. Každý objekt musí mít nejméně jeden identifikátor. Musí platit  $D \supseteq \bigcup I$ . Pro ilustraci – při převodu z ER bude pro původní entitní typy a atributy platit  $D = \bigcup I$ , ale u vztahových typů může být v  $D$  něco navíc.

Morfismus je osmice (signatura, doména, kodoména, směr, název, kardinalita, duplicity, uspořádání). Signatura vyjadřuje „cestu“. Jedná se o řetězec složený ze signatur jednotlivých morfismů, z kterých se morfismus skládá. Pokud se z dalších morfismů neskládá, je signatura unikátní identitou morfismu (symbol). Pro ilustraci viz obrázek 2.2.

Doména a kodoména jsou identity příslušných objektů, odpovídají tak zdrojovému a cílovému objektu kategorie.

Směr udává směr morfismu (který nemusí nutně odpovídat dvojici doména/kodoména). To proto, že pro každý morfismus bude existovat jeho protějšek. Směr má dvě možné hodnoty a budeme je značit 1 (tam), 0 (zpět).

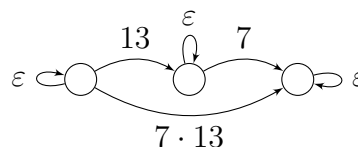
Kardinalita odpovídá tomu, jak byla popsána pro ER v sekci 2.1 na straně 16. Při skládání se kardinality transformují následovně: pro kardinality  $(a, b)$  a  $(c, d)$  je jejich složením  $(\min(a, c), \max(b, d))$ . Uspořádání nad symboly kardinalit je intuitivně  $0 < 1 < *$ .

Duplicity a uspořádání jsou booleovské hodnoty, které odpovídají tomu, jestli jsou povoleny tyto jevy. Tyto hodnoty mají význam pouze pokud je maximální kardinalita  $*$ .

Potřebuji dovysvětlit.

Morfismy ve schematické kategorii rozdělíme na několik druhů, které jsou všechny vidět na obrázku 2.2:

- *bázové* (base) morfismy jsou ty, které odpovídají jednotlivým spojením a ISA hierarchiím mezi objekty; signatura je identita,
- *identitní* (identity) morfismy jsou ty, které vznikly kvůli axiomu identity; signatura je  $\varepsilon$ ,
- *odvozené* (composite) morfismy jsou ty, které vznikly kvůli tranzitivitě; signatura je opravdová cesta (zřetězené signatury); zřetězování zapisujeme ve stejném pořadí jako skládání morfismů, aby bylo analogické.



Obrázek 2.2: Signatury morfismů,  $\cdot$  je operace konkatence (zřetězování) symbolů

## 3. Specifikace

Sommerville [19] uvádí fundamentální aktivity softwarového inženýrství: specifikace, vývoj, validace a evoluce. Dále definuje softwarovou specifikaci jako aktivitu, při které zákazníci a inženýři definují software, který má být vyprodukován, a omezení na jeho provoz.

V této kapitole definujeme software, který budeme tvořit, pomocí požadavků uživatele, případů užití, diagramu tříd v konceptuální vrstvě a procesů.

### 3.1 Požadavky

Požadavky na softwarový systém jsou popisy toho, co by měl systém dělat – služby, které poskytuje, a omezení na jeho provoz. Tyto požadavky by měly reflektovat potřeby zákazníka na systém a jeho účel, uvádí Sommerville [19, s. 83].

Požadavky na funkce systému nazýváme funkční. Požadavky na provoz systému a jeho omezení nazýváme nefunkční.

Dále uvedeme vybrané funkční a nefunkční požadavky z pohledu uživatele systému.

#### 3.1.1 Funkční požadavky

##### Projekt

**FP-1** V systému bude možné vytvořit nový projekt.

**FP-2** Projekt bude možné uložit.

**FP-3** Projekt bude možné načíst.

**FP-4** Projekt bude možné pojmenovat pro odlišení od ostatních projektů.

**FP-5** Jednotlivé diagramy bude možné exportovat do rastrového i vektorového formátu.

**FP-6** Do těchto exportovaných formátů bude volitelně možné vložit projekt, který z nich pak bude možné načíst. Tímto bude projekt možné otevřít jak v prohlížeči obrázků (a zobrazit rastrově nebo vektorově diagram), tak v našem systému a pokračovat v práci.

**FP-7** Zobrazení každého diagramu bude možné posouvat myší.

**FP-8** Zobrazení každého diagramu bude možné přibližovat a oddalovat kolečkem myši.

**FP-9** Posunutí a přiblížení bude volitelně možné synchronizovat mezi všemi diagramy.

**FP-10** Systém bude kontrolovat validitu uživatelem vytvořených konstruktů.



## ER Diagram

- FP-11** Do diagramu bude možné přidat entitní typ.
- FP-12** Do diagramu bude možné přidat vztahový typ.
- FP-13** Do diagramu bude možné přidat atribut.
- FP-14** Mezi entitním a vztahovým typem bude možné vytvořit spojení.
- FP-15** Mezi entitním typem a atributem bude možné vytvořit spojení.
- FP-16** U spojení bude možné specifikovat a zobrazit kardinalitu. Dolní mez bude buď 0 nebo 1, horní mez 1 nebo  $n$ . Výchozí kardinality (1..1) nebudou zobrazeny.
- FP-17** Bude možné vytvořit ISA hierarchii mezi entitními typy.
- FP-18** Vlastnosti všech objektů bude možné měnit (popisky, typ, pozice). Tyto změny budou reflektovány v ostatních diagramech.
- FP-19** Objekty bude možné posunovat držením levého tlačítka myši a tažením.
- FP-20** Všechny objekty bude možné zvolit levým tlačítkem myši.
- FP-21** Při držení klávesy Ctrl bude možné zvolit více objektů najednou postupným klikáním levého tlačítka myši.
- FP-22** Veškeré objekty bude možné z diagramu mazat alespoň klávesou Delete.
- FP-23** Uživatel bude moci využít předpřipravené konstrukty, které bude možné vložit do diagramu. Například se může jednat o ISA hierarchie s předpřipravenými entitami.

## Schématická kategorie

- FP-24** Objekty a morfismy schématické kategorie bude možné přidávat a mazat, přičemž tato změna se odrazí v ostatních diagramech.
- FP-25** V diagramu se zobrazí data o objektech a morfismech včetně kardinalit a popisků. Výchozí kardinality (1..1) se zobrazovat nebudou.
- FP-26** Objekty bude možné posouvat a jejich posunutí se odrazí na ostatních diagramech.
- FP-27** Při zvolení objektu se barevně vyznačí všechny objekty v okolí, které ho identifikují (a jinou barvou identifikátory vzdálené).

## Vizualizace schématické kategorie

- FP-28** Vizualizace schématické kategorie ze schématické kategorie vytěží sémantiku a vizualizuje ji různými tvary.
- FP-29** Při zvolení objektu se barevně zvýrazní bezprostřední a vzdálené identifikátory různými barvami.

### 3.1.2 Nefunkční požadavky

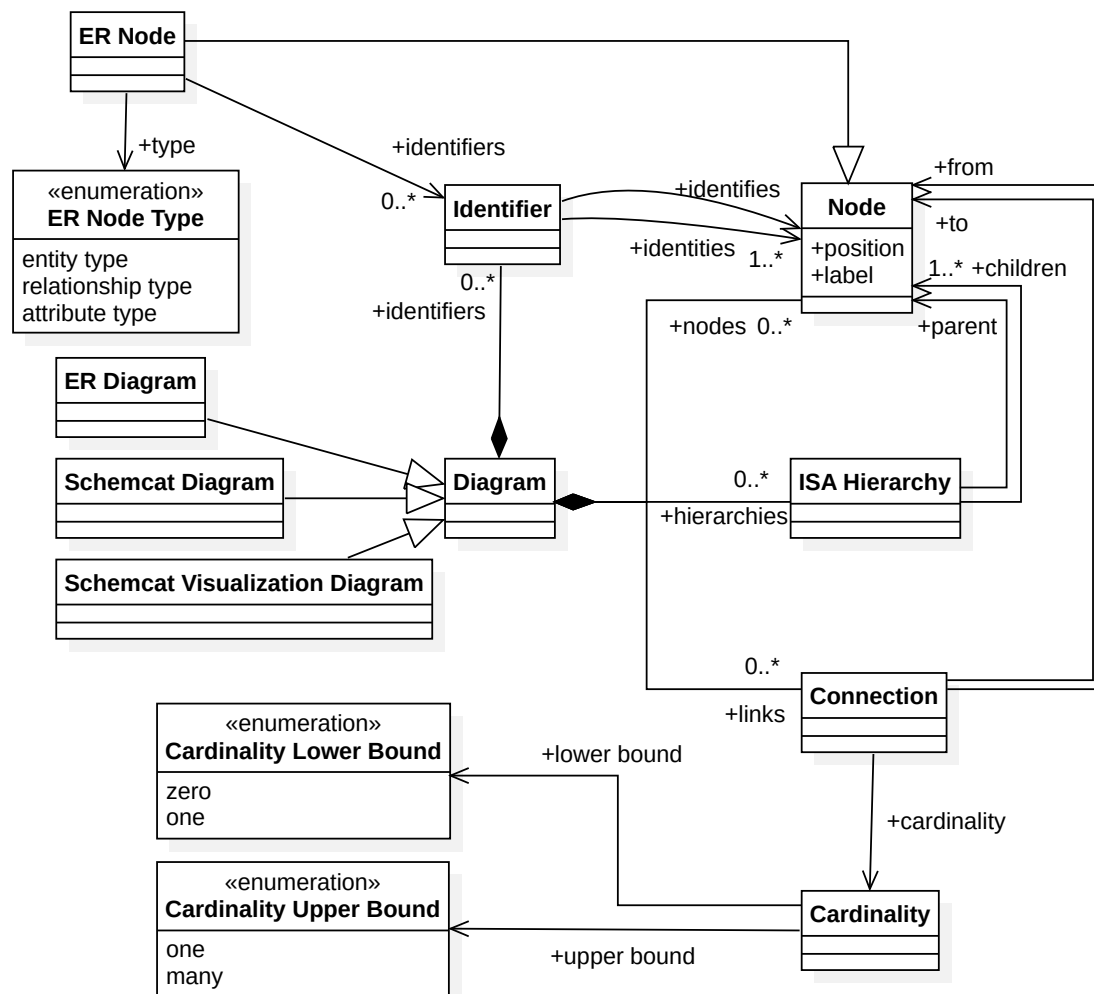
- NFP-1** Aplikaci bude možné používat na všech běžných desktopových operačních systémech.
- NFP-2** Nesmí dojít ke ztrátě práce při náhlém ukončení aplikace kvůli interním či externím vlivům. To může být zařízeno např. průběžným ukládáním práce.
- NFP-3** Aplikaci bude možné používat i při výpadku internetového připojení.
- NFP-4** V rámci bezpečnosti žádná data týkající se práce na projektu neopustí zařízení klienta, pokud tak klient explicitně neučiní (například export a přesun souboru).
- NFP-5** Aplikace bude navržena tak, aby bylo možné bez větších komplikací rozšířit její funkcionalitu (např. přidat podporu UML).

## 3.2 Diagram tříd

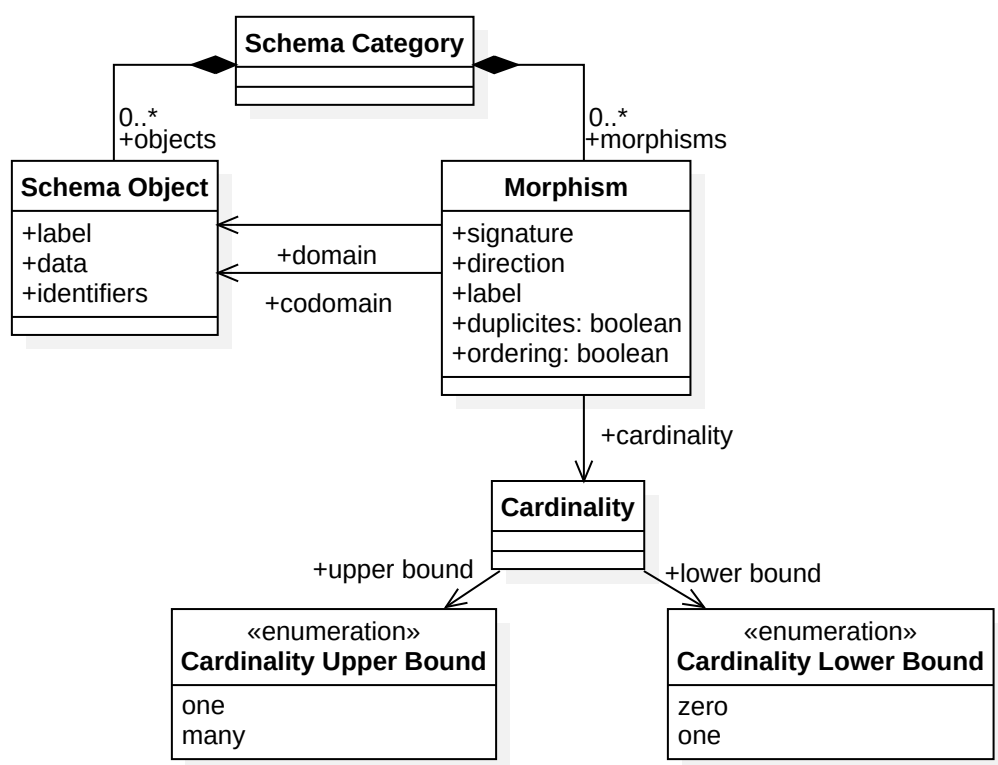
V této kapitole představíme konceptuální model. Jedná se o část světa, na kterou vymezujeme svůj diskurz.

Na obrázku 3.1 je UML diagram tříd konceptuálního modelu diagramů.

Diagram konceptuálního modelu schematické kategorie lze pozorovat na obrázku 3.2.



Obrázek 3.1: Diagram tříd – diagramy



Obrázek 3.2: Diagram tříd – schematická kategorie

# Závěr

Závěr

# Seznam použité literatury

- [1] JGraph Ltd. diagrams.net. [online]. URL: <https://www.diagrams.net/>.
- [2] DrawSQL. DrawSQL – Database Schema Diagrams. [online]. URL: <https://drawsql.app/>.
- [3] ERDPlus. ERDPlus. [online], 2021. URL: <https://erdplus.com/>.
- [4] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, březn 1976. doi:10.1145/320434.320440.
- [5] Nomnoml. URL: <https://www.nomnoml.com>.
- [6] Object Management Group (OMG). Unified Modeling Language version 2.5.1. [online], prosinec 2017. URL: <https://www.omg.org/spec/UML/>.
- [7] Visual Paradigm Online - Suite of Powerful Tools. URL: <https://online.visual-paradigm.com/>.
- [8] Schulze Dirk and SVG Working Group. Scalable Vector Graphics (SVG) 2 Specification. W3C Recommendation CR-SVG2-20181004, W3C, říjen 2018. URL: <https://www.w3.org/TR/2018/CR-SVG2-20181004/>.
- [9] drawio on GitHub – Open-source, not open-contribution, srpen 2022. URL: <https://github.com/jgraph/drawio/blob/c7122cad617f52563c11d90890e64ab06db3a27a/README.md#open-source-not-open-contribution>.
- [10] OpenJS Foundation. Electron. [online], 2021. URL: <https://www.electronjs.org/>.
- [11] Martin Seibert. Extracting the XML from mxfiles, srpen 2016. URL: <https://drawio-app.com/extracting-the-xml-from-mxfiles/>.
- [12] MathJax Consortium. MathJax. URL: <https://www.mathjax.org/>.
- [13] TC39 Group. The JSON data interchange syntax 2nd Edition. Standard ECMA-404, ECMA International, Rue du Rhône 114, CH-1204 Ženeva, Švýcarsko, prosinec 2017. URL: <https://www.ecma-international.org/publications-and-standards/standards/ecma-404/>.
- [14] Facebook Inc. React, A JavaScript library for building user interfaces. [online], 2021. URL: <https://reactjs.org/>.
- [15] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, březn 1976. URL: <https://dl.acm.org/doi/10.1145/320434.320440>, doi:10.1145/320434.320440.
- [16] Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, and Riccardo Torlone. *Database systems: concepts, languages & architectures*. McGraw-Hill, New York, 1999.

- [17] Samuel Eilenberg and Saunders MacLane. General theory of natural equivalences. *Transactions of the American Mathematical Society*, 58(0):231–294, 1945. URL: <https://www.ams.org/tran/1945-058-00/S0002-9947-1945-0013131-6/>, doi:10.1090/S0002-9947-1945-0013131-6.
- [18] Martin Svoboda, Pavel Čontoš, and Irena Holubová. Categorical Modeling of Multi-model Data: One Model to Rule Them All. In Christian Attiogbé and Sadok Ben Yahia, editors, *Model and Data Engineering*, Lecture Notes in Computer Science, pages 190–198, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-78428-7\_15.
- [19] Ian Sommerville. *Software engineering*. Pearson, 9th ed edition, 2011.

# Seznam obrázků

1.1	Tvorba ER diagramu v aplikaci diagrams.net . . . . .	8
1.2	Tvorba diagramu v drawSQL . . . . .	10
1.3	Tvorba ER diagramu v ERDplus . . . . .	11
1.4	Tvorba UML diagramu v nomnoml . . . . .	13
2.1	Příklad kategorie . . . . .	18
2.2	Signatury morfismů, $\cdot$ je operace konkatence (zřetězování) symbolů	19
3.1	Diagram tříd – diagramy . . . . .	23
3.2	Diagram tříd – schematická kategorie . . . . .	24



# Seznam tabulek

1.1	Srovnání existujících řešení . . . . .	15
2.1	Grafická reprezentace konstruktů ER modelu . . . . .	17