
F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

# Automatické vyhodnocování úloh v předmětu Databázové systémy

**Valeriia Klimova**

**Květen 2023**
**Program: Otevřená Informatika**
**Specializace: Software**
**Vedoucí práce: RNDr. Ingrid Nagyová, Ph.D.**

# BACHELOR‘S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Klimova Valeriia** Personal ID number: **499008**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Software**

## II. Bachelor’s thesis details

Bachelor’s thesis title in English:

**Automatic evaluation of tasks in the subject Database systems**

Bachelor’s thesis title in Czech:

**Automatické vyhodnocování úloh v předmětu Databázové systémy**

Guidelines:

Bibliography / sources:

Svoboda, Martin. 2021. „Přednášky z předmětu Databázové systémy.“ Dostupné 31. led. 2023 https://www.ksi.mff.cuni.cz/~svoboda/courses/182-B0B36DBS/
Pokorný, Jaroslav. 2020. „Databázové systémy.“ Praha: Nakladatelství ČVUT.
Farana, Radim. 2004. „Tvorba relačních databázových systémů.“ Ostrava: VŠB-TU.
Oracle and/or its affiliates. 2023. „Database SQL Reference.” Dostupné 31. led. 2023 https://docs.oracle.com/cd/B19306_01/server.102/b14200/toc.htm

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: **09.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

| RNDr. Ingrid Nagyová, Ph.D.<br>Supervisor’s signature | Head of department’s signature | prof. Mgr. Petr Páta, Ph.D.<br>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

# Poděkování / Prohlášení

Ráda bych poděkovala vedoucí mé bakalářské práce RNDr. Ingrid Nagyové, Ph.D. za její odborné vedení, cenné rady a přátelský přístup, což mi výrazně pomohlo při dosahování cílů tohoto projektu. Dále bych chtěla poděkovat svým rodičům a blízkým přátelům za veškerou podporu, kterou mi poskytovali a která mi dodávala sílu a motivaci po celou dobu mého studia.

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

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

.....

# Abstrakt / Abstract

Tato práce se zaměřuje na automatizaci vyhodnocení úlohy v předmětu Databázové systémy, která spočívá v převodu relačního schématu do SQL schématu pro vytvoření odpovídajících tabulek v existujícím databázovém systému. Obsahuje podrobný úvod do úlohy, analýzu možností její automatické evaluace, zhodnocení stávajících nástrojů, návrh vlastního nástroje pro automatické vyhodnocení úlohy, popis jeho implementace a následného testování.

Hodnotící nástroj byl implementován v jazyce Java s využitím H2 databáze. Běh programu je rozdělen do čtyř fází: inicializace, parsování, kontrola a výpočet bodů, a generování výsledku. Program byl testován na 81 studentských pracích a správně rozpoznává splnění kritérií úlohy ve více než 75 % případů.

Po integraci do systému BRUTE bude hodnotící nástroj fungovat tak, že nahraje student své řešení úlohy, nástroj provede kontrolu a navrhne počet bodů podle stanoveného rozložení. Pokud je student spokojen se svým hodnocením a nebude provádět žádné další úpravy, učitel manuálně zkontroluje zbývající části zadání a přidělí finální počet bodů.

**Klíčová slova:** relační model; SQL; syntaxe dotazů; integritní omezení; automatické vyhodnocování; zpětná vazba.

The main topic of this thesis is the automation of task evaluation in the Database Systems course. The task involves translating a relational schema into an SQL schema and creating corresponding tables in a database system. The thesis presents a detailed overview of the task, an analysis of possible automated evaluation methods, existing tools, a custom tool design, implementation details, and testing results.

The evaluation tool was implemented in Java using the H2 database. The execution of the program is divided into four phases: initialization, parsing, checking and calculating points, and generating the result. It has been tested on 81 student solutions and successfully identifies the fulfillment of the task criteria in over 75% of the cases.

Once integrated into the BRUTE system, the evaluation tool will function as follows: the student will upload a solution, and the tool will perform a thorough check and suggest a score based on a predefined distribution. If the student accepts the score and does not modify the solution, the teacher will manually review the remaining parts and assign the final score to the student.

**Keywords:** relational model; SQL; query syntax; integrity constraints; automatic evaluation; feedback.

**Title translation:** Automatic evaluation of tasks in the subject Database systems

# / Obsah

# Tabulky / Obrázky

# Kapitola 1
# Úvod

Automatizace procesů v mnoha různých oblastech života umožňuje provádět činnosti bez přímého zásahu člověka [1]. Použití této technologie může mít řadu výhod, například urychlení těchto procesů, aniž by to bylo na úkor důkladnosti a detailnosti práce. Nejdůležitější rozhodnutí a kontrola nad výsledky prováděných činností jsou však stále ponechány na lidech.

Automatizace našla své uplatnění také v oblasti vzdělávání. Jedním z běžných způsobů jejího použití je hodnocení domácích úkolů studentů. Tato práce se zabývá konkrétně vytvořením nástroje pro kontrolu úlohy v předmětu Databázové systémy, která spočívá v překladu relačního schématu do schématu jazyka SQL pro vytvoření odpovídajících tabulek v existujícím databázovém systému.

Cílem projektu je seznámit se s úlohou zadávanou v předmětu Databázové systémy, analyzovat možnosti její automatické evaluace a navrhnout řešení problému automatického vyhodnocování. Následně je třeba navržený hodnotící nástroj implementovat a otestovat ho kontrolou skutečných příkladů řešení úlohy.

## 1.1 Definice základních pojmů

Před provedením analýzy samotné úlohy je nezbytné se seznámit se základními pojmy a komponentami jednotlivých částí zadání.

### 1.1.1 Relační model

Existuje řada způsobů, jak popsat, co se relačním modelem rozumí. Níže jsou uvedeny nejvýznamnější definice pro první setkání studentů s databázovými systémy:

- Dle [2] je relační model stylem modelování založeným na matematické struktuře zvané relace. Každou relaci lze představit jako tabulku. Reprezentace dat pomocí její řádků pak znamená navrhovaní vztahu mezi daty.
- Dle [3] je relační model způsobem modelování logických a matematických datových struktur založený na relacích mezi entitami, kde každé entitě odpovídá tabulka v databázi.

Pro práci s relačním modelem však nestačí znát definici. K hlubšímu porozumění problematiky je nezbytné pochopit komponenty, které tvoří relační model.

Relační model je v podstatě jeden ze způsobu datového modelování, jehož základními prvky jsou entity, vztahy a atributy. Z názvu relačního modelu plyne, že popisuje vztahy mezi jednotlivými entitami reálného světa. Je také těsně spojen s pojmem relace v matematice. Přesněji, v relačním modelu "se datové struktury modelují pomocí matematických relaci popsaných schématem relace" [2].

*Schéma relace* má tvar R($A_1$:$D_1$, $A_2$:$D_2$, ..., $A_n$:$D_n$), kde:

- R je samotná relace,
- n $\geq$1 je arita relace,

- $A_i$, 1 ≤i ≤n je množina názvů atributů relace,
- $D_i$, 1 ≤i ≤n je množina domén relace, které popisují, jak jsou reprezentována data v počítači.

Množina atributů, která jednoznačně identifikuje prvky v relaci nad schématem R, se nazývá *primární klíč* [4]. Základním požadavkem na primární klíč je absence v této množině redundantních atributů. Značí se obvykle podtržením. Důležitým pojmem je také *cizí klíč*, což je projekce nějaké množiny atributů na hodnoty atributů jiné entity [5]. Oba tyto pojmy patří k *integritním omezením* pro relaci R.

Entita reprezentující osobu s atributy jméno, příjmení, datum narození, rezidence, pohlaví a email by mohla být zapsána v relačním schématu například takto: `Person (firstname, surname, dateOfBirth, residence, gender, email)`, kde množina atributů `(firstname, surname, dateOfBirth, residence)` je primárním klíčem.

Z entity Person je možné zdědit novou entitu reprezentující zaměstnance nějaké společnosti: `Employee (firstname, surname, dateOfBirth, residence, position, company)`, kde množina atributů `(firstname, surname, dateOfBirth, residence)` je příkladem cizího klíče, který je v tomoto případě současně primárním klíčem, značí se jako `FK: (firstName, surname, dateOfBirth, residence)` ⊆ `Person (firstName, surname, dateOfBirth, residence)`.

### 1.1.2 Dotazovací jazyk SQL

Použijme dva nezávislé zdroje k formulaci a lepšímu pochopení toho, co je myšleno dotazovacím jazykem SQL:

- Dle [2] SQL je jazykem relačních databází založeným na relačním datovém modelu, jednotlivým relacím se však říká tabulky. Používá se pro práci s daty, dovoluje vytvářet tabulky, plnit je daty a pak data vybírat podle požadavků, neboli dotazů. Oproti relačnímu modelu SQL povoluje používání prázdných hodnot v tabulkách a více typů integritních omezení. Výhodou jazyka SQL je nezávislost na fyzické organizaci dat. Na rozdíl od procedurálních programovacích jazyků, které určují jakým způsobem má být sekvence příkazů provedena, popisuje, co je od databáze požadováno.
- Dle [3] SQL je dotazovací jazyk široce používaný v relačních databázových systémech, hlavně za účelem dotazování a manipulace s daty, ale také pro vytvoření schématu.

Data se v SQL deklarují jako záhlaví dvourozměrné tabulky se sloupci odpovídajícími jednotlivým atributům relačního modelu pomocí příkazu `CREATE TABLE`. Vytvoření tabulky pro osobu ze sekce 1.1.1 by mohlo vypadat například jako na obrázku 1.1.

```
CREATE TABLE IF NOT EXISTS Person (
    person_ID SERIAL PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    dateOfBirth DATE NOT NULL,
    residence VARCHAR(200) NOT NULL,
    gender VARCHAR(10) NOT NULL,
    email VARCHAR(100) NOT NULL CHECK (email LIKE '_%@_%.__%'),
    UNIQUE (firstName, lastName, dateOfBirth, residence)
);
```

**Obrázek 1.1.** Příklad vytvoření tabulky pro osobu v jazyku SQL

Tabulka pro zaměstnance ze sekce 1.1.1 s použitím referenčních omezení, neboli cizích klíčů, by se pak vytvářela jako na obrázku 1.2.

```
CREATE TABLE IF NOT EXISTS Employee (
    employee_ID INTEGER PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    dateOfBirth DATE NOT NULL,
    residence VARCHAR(200) NOT NULL,
    position VARCHAR(50) NOT NULL,
    company VARCHAR(50) NOT NULL,
    UNIQUE (firstName, lastName, dateOfBirth, residence),
    CONSTRAINT Employee_fk_Person_ID
        FOREIGN KEY (employee_ID)
        REFERENCES Person (person_ID)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Employee_fk_Person
        FOREIGN KEY (firstName, lastName, dateOfBirth, residence)
        REFERENCES Person (firstName, lastName, dateOfBirth, residence)
        ON UPDATE CASCADE ON DELETE CASCADE
);
```

**Obrázek 1.2.** Příklad vytvoření tabulky pro zaměstnance v jazyku SQL

Je vidět, že oproti relačnímu modelu SQL dovoluje používat uměle klíče. Jsou to primární klíče vytvořené z automaticky generované sekvence čísel [6]. Výhodou je přehlednější struktura tabulky, což se pak hodí pro snadnější vyhledávání v databázi, především při spojení tabulek pomocí příkazu `JOIN`. SQL také umožňuje nastavovat integritní omezení nejen na úrovni atributu, ale i na úrovni celé tabulky.

## 1.2 Seznámení s úlohou

Studenti mají sestavený v předešlých úkolech relační model reprezentující zvolené téma a popisující strukturu budoucí databáze. Táto práce se zaměří na úlohu, jejímž cílem je formulace SQL dotazů vytvářejících tabulky v databázi na základě tohoto modelu a naplnění tabulek daty, přičemž jedna z nich bude obsahovat minimálně 32 tisíc dat. Součástí zadání je zavedení:

- atributových integritních omezení,
- tabulkových integritních omezení,
- cizích klíčů včetně direktivy `ON UPDATE/DELETE`,
- primárních nebo, pokud je to vhodné, umělých klíčů.

Pak je nutné nad touto databází formulovat dotazy pokrývající alespoň jednou:

- vnější spojení tabulek,
- vnitřní spojení tabulek,
- podmínku na data,
- agregaci a podmínku na hodnotu agregační funkce,
- řazení a stránkování,
- množinové operace,
- vnořený `SELECT`.

Relační model a SQL dotazy se odevzdávají ve formě PDF souborů s vysvětlujícími poznámkami pro učitele, kvůli tomu musí nástroj pro automatickou kontrolu umět pracovat s tímto formátem a následně načítat potřebné informace pro analýzu. V ideálním případě by měla být aplikace schopna ověřit splnění požadavků zadání a porovnat, zda navržené řešení odpovídá dříve vytvořenému modelu.

# Kapitola 2
# Analýza úlohy

Aby mohl hodnotící nástroj analyzovat a bodovat práce studentů, je třeba zjistit, jak jsou přidělovány body za jednotlivé části úlohy, jakých chyb se studenti často dopouštějí a jak jejich práce hodnotí učitelé předmětu Databázové systémy.

## 2.1 Hodnocení

Ačkoli neexistuje žádný oficiální způsob rozdělení bodů za jednotlivé části úkolu, je to nezbytným požadavkem pro vytvoření hodnotícího nástroje. Podle znění zadání je logické rozdělit hodnocení úlohy na tři části: tvorba tabulek v databázi, naplnění tabulek daty a formulace dotazů nad databází.

Při tvorbě tabulek může student získat 10 bodů podle následujících kritérií:

- tabulky odpovídající relačnímu modelu – 5 bodů,
- definování primárních klíčů tabulek – 1 bod,
- využití atributového integritního omezení – 1 bod,
- využití tabulkového integritního omezení – 1 bod,
- využití `ON UPDATE/DELETE` u cizího klíče – 2 body.

Jakmile student vytvoří tabulky, je potřeba je naplnit daty. Zde může nastat problém s kontrolou, když nebude poskytnut přístup do školního databázového systému. Za naplnění tabulek může student získat maximálně 5 bodů:

- příkazy pro naplnění tabulek – 2 body,
- dostupnost tabulek na serveru – 1 bod,
- naplnění alespoň jedné tabulky větším počtem dat – 2 body.

Za poslední a nejrozsáhlejší část úlohy může student získat 15 bodů. Jejím cílem je napsání logicky správných a smysluplných dotazů, které pokrývají požadavky uvedené v zadání a obsahují následující prvky:

- vnější spojení tabulek – 2 body,
- vnitřní spojení tabulek – 2 body,
- podmínku na data – 1 bod,
- agregaci a podmínku na hodnotu agregační funkce – 2 body,
- řazení a stránkování – 1 bod,
- množinové operace – 1 bod,
- vnořený `SELECT` – 2 body,
- SQL dotazy bez chyb a popis dotazů – 4 body.

Celkový počet bodů za řešení úlohy je 30 bodů, pokud jsou splněna všechna kritéria.

## 2.2 Analýza prací studentů

Tato část analýzy vychází z řešení 20 studentů v rámci předmětu Databázové systémy v letním semestru akademického roku 2021/2022. Řešení každého studenta obsahovalo soubor s relačním modelem a soubor s jeho převodem do jazyka SQL.

Chyby, které studenti dělají při transformaci relačního modelů do SQL tabulek a při vytvoření dotazů nad nimi, lze rozdělit do několika kategorií. Za prvé, často se vyskytují chyby tykající se vytvoření atributů nebo integritních omezení. Za druhé, studenti někdy zapomínají nebo nezvládají použit požadované příkazy nebo operace. Seznam takových chyb doplněný informací, v kolika pracích se každá z nich vyskytla, je v tabulce 2.1.

Bude se také hodit uvést seznam nebezpečných míst v řešeních studentů, která by mohla způsobit problémy při implementaci hodnotícího nástroje a kterým je vhodné věnovat pozornost. Zjištěná rizika jsou také vypsána v tabulce 2.1 a doplněna sloupcem s počtem prací, ve kterých se každé z nich vyskytlo.

| Popis | Počet výskytů |
|---|---|
| **Chyby při použití integritních omezení a tvorbě atributů** | |
| Umělý klíč není správně zaveden nebo použit | 14 krát |
| Direktiva `NOT NULL` a/nebo `UNIQUE` je zbytečně použita u primárního klíče | 7 krát |
| Chybí primární klíč v tabulce | 7 krát |
| Cizí klíč je použit bez direktivy `ON DELETE/UPDATE` | 3 krát |
| Atributové integritní omezení je použito namísto jednoho tabulkového | 3 krát |
| Je použit nevhodný typ pro datum, např. narození nebo nástup do práce | 2 krát |
| Cizí klíč není správně použit, např. je rozdělen na menší částí | 2 krát |
| Cizí klíč odkazuje na neexistující tabulku | 1 krát |
| **Chyby v příkazech nebo operacích** | |
| Chybí příkazy pro naplnění tabulek | 13 krát |
| Chybí `DROP TABLE IF EXISTS` při vytvoření tabulek | 8 krát |
| Vnější spojení v dotazech není použito | 3 krát |
| Množinové operace v dotazech nejsou použity | 3 krát |
| Agregace v dotazech není použita | 2 krát |
| **Rizika při implementaci** | |
| Počet atributů je odlišný oproti relačnímu modelu | 9 krát |
| Názvy tabulek jsou odlišné oproti relačnímu modelu | 6 krát |
| Je použit příkaz `ALTER TABLE` | 5 krát |
| Počet tabulek je odlišný oproti relačnímu modelu | 4 krát |
| Několik tabulek z relačního modelu je spojeno do jedné tabulky | 3 krát |
| Integritní omezení neodpovídá relačnímu modelu | 1 krát |
| Jsou mezery v názvech v relačním modelu | 1 krát |

**Tabulka 2.1.** Chyby vyskytující se v pracích studentů

## 2.3 Ruční kontrola prací

Kromě prohlížení prací studentů byly navíc analyzovány komentáře učitelů při ručním hodnocení 116 řešení úlohy tykající se práce s SQL databází z předmětu Databázové systémy v letním semestru akademického roku 2021/2022. Výsledkem průzkumu je seznam nalezených chyb v tabulce 2.2, které jsou opět rozděleny do jednotlivých kategorií: při použití integritních omezení nebo tvorbě atributů, v příkazech nebo operacích a zbývající chyby, které se nepodařilo zařadit do již existujících kategorií. Každá chyba má spočítáno, kolikrát byla v komentářích zmíněna.

| Popis | Počet výskytů |
|---|---|
| **Chyby při použití integritních omezení a tvorbě atributů** | |
| Umělý klíč není správně zaveden nebo použit | 22 krát |
| Chybí primární klíč v tabulce | 20 krát |
| Chybí atributové integritní omezení | 14 krát |
| Chybí odkaz cizího klíče do své tabulky | 13 krát |
| Cizí klíč je použit bez direktivy `ON DELETE/UPDATE` | 10 krát |
| Atributy v tabulkách neodpovídají relačnímu modelu | 7 krát |
| Chybí tabulka s dvěma nezávislými primárními klíči | 5 krát |
| Chybí složené a vícenásobné klíče | 4 krát |
| Rozdělení cizího klíče do několika integritních omezení | 3 krát |
| **Chyby v příkazech nebo operacích** | |
| Chybí minimálně jeden z požadovaných dotazů | 41 krát |
| Chybí použití `HAVING` | 39 krát |
| Direktiva `ORDER BY` je použita bez `LIMIT` a/nebo `OFFSET` | 34 krát |
| Chybí tabulky z relačního modelu | 13 krát |
| Chybí použití `GROUP BY` | 12 krát |
| Chybí dotazy pro vytvoření databáze | 5 krát |
| Vnořený select nevrací množinu | 2 krát |
| **Ostatní chyby** | |
| Je logická chyba v dotazu | 41 krát |
| Chybí tabulka s 32K řádky | 17 krát |
| Odevzdány jsou screenshoty dotazů, není je možné zpracovat | 5 krát |
| Databáze je prázdná | 1 krát |

**Tabulka 2.2.** Chyby zmíněné v hodnoceních prací učiteli

## 2.4 Shrnutí

Chyby uvedené v sekcích 2.2 a 2.3 lze spojit do jedné tabulky 2.3, která je pro přehlednost opět rozdělena do jednotlivých kategorií, přičemž každá chyba má unikátní kód.

Ze seznamu chyb je vhodné upozornit na ty, které vyžadují ruční kontrolu. Za prvé je to logická složka úlohy. Za druhé je to část zadání, pro kterou je potřeba mít přístup k databázi. Na základě rozdělení bodů ze sekce 2.1 lze spočítat, že student nemůže bez kontroly učitele dostat 7 ze 30 bodů:

- dostupnost tabulek na serveru – 1 bod,
- naplnění alespoň jedné tabulky větším počtem dat – 2 body,
- SQL dotazy bez chyb a popis dotazů – 4 body.

| Popis | Kód chyby |
|---|---|
| **Chyby při použití integritních omezení a tvorbě atributů** | |
| Nesprávně zvolený typ pro umělé klíče | A-1 |
| Neexistence nebo nesprávné použití primárního klíče v tabulce | A-2 |
| Nesprávné použití cizích klíčů | A-3 |
| Cizí klíče bez `ON DELETE/UPDATE` | A-4 |
| Nesprávné použití integritních omezení | A-5 |
| Nevhodný typ pro datum, např. narození nebo nástup do práce | A-6 |
| **Chyby v příkazech nebo operacích** | |
| Chybí příkazy pro naplnění tabulek | B-1 |
| Chybí `DROP TABLE IF EXISTS` při vytvoření tabulek | B-2 |
| Chybí minimálně jeden z požadovaných dotazů | B-3 |
| **Rizika při implementaci** | |
| Odevzdány jsou screenshoty dotazů, není je možné zpracovat | C-1 |
| Použití `ALTER TABLE` | C-2 |
| Atributy v tabulkách neodpovídají relačnímu modelu | C-3 |
| Přejmenovaní tabulek, převod názvu z češtiny do angličtiny | C-4 |
| Přidání nebo odstranění tabulek oproti relačnímu modelu | C-5 |
| Spojení několika tabulek z relačního modelu | C-6 |
| Mezery v názvech v relačním modelu | C-7 |
| **Ostatní chyby** | |
| Logická chyba v dotazu | D-1 |
| Chybí tabulka s 32K řádky | D-2 |
| Databáze je prázdná | D-3 |

**Tabulka 2.3.** Chyby vyskytující se v pracích studentů

Teoreticky může student po automatické evaluaci získat maximálně 23 bodů, pokud jeho řešení splní následující požadavky:

- tabulky odpovídají relačnímu modelu – 5 bodů,
- využívá integritní omezení, včetně primárních klíčů u všech tabulek a direktivy `ON UPDATE/DELETE` u cizích klíčů – 5 bodů,
- obsahuje příkazy pro naplnění tabulek – 2 body,
- splňuje kritéria pro dotazy – 11 bodů.

Implementace načtení relačního modelu ze souboru, rozpoznání popisů jednotlivých tabulek a jejich analýzy může být příliš časově náročná, takže pokud budou kontrolována pouze základní kritéria zadání týkající se obsahu dotazů, získá student maximálně 18 bodů.

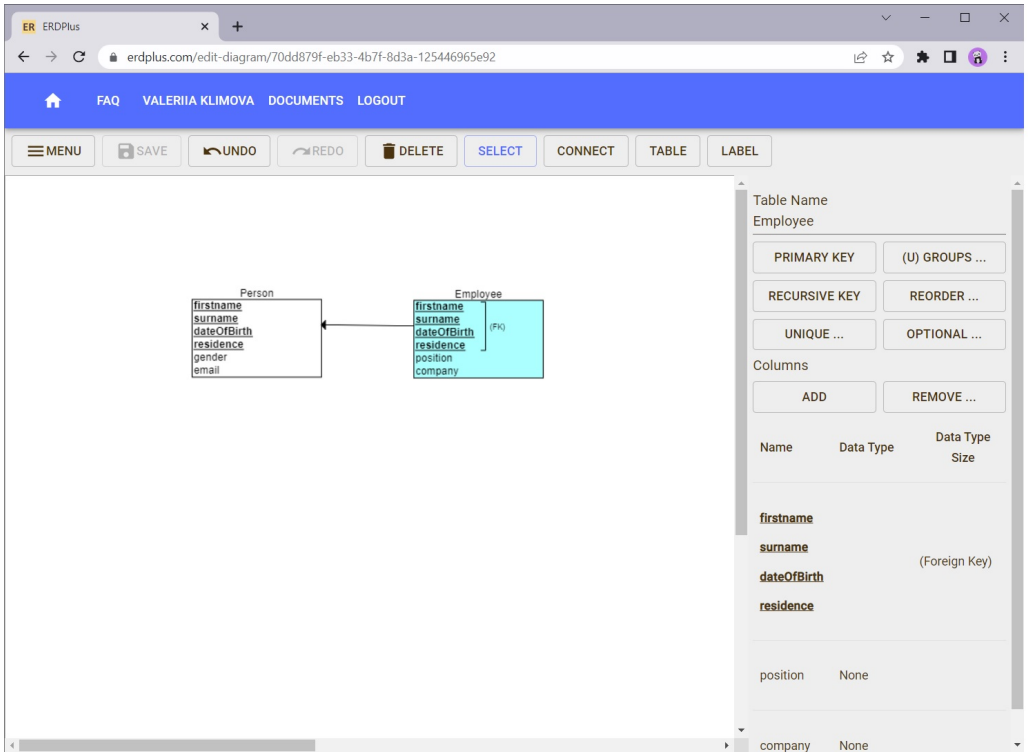
Největším rizikem při implementaci nástroje pro automatické vyhodnocení je použití příkazu `ALTER TABLE` pro přidání integritních omezení. Dalším rizikem v případě práce s relačním modelem je výrazný rozdíl mezi tabulkami v databázi a původním relačním modelem. Nástroj nebude vůbec fungovat, pokud student nahraje své řešení ve formě snímků obrazovky, a nikoli v textové podobě.

# Kapitola 3
# Analýza existujících nástrojů

Implementaci hodnotícího nástroje lze rozdělit na dvě velké části: kontrolu plnění kritérií na obsah dotazů a analýzu výsledného schématu v databázi ve srovnání s relačním schématem. Řešením pro realizaci druhé části je převedení relačního modelu na SQL dotazy a porovnání vygenerovaných dotazů se studentovými. Pro posouzení reálnosti a složitosti vytvoření takového překladače je nutné prostudovat existující nástroje se shodným účelem, které by mohly sloužit jako inspirace a zdroj nápadů, jak by se to mohlo udělat správně.

## 3.1 Převod relačního modelu na SQL dotazy

Webový nástroj ERDPLus[1] umožňuje vytvořit relační model graficky ve formě diagramu. Tabulky pro osobu a zaměstnance ze sekce 1.1.1 je možné znázornit jako na obrázku 3.1. Na rozdíl od požadavků na úlohu v předmětu Databázové systémy se dají nastavit typy atributů a kromě primárních a cizích klíčů také další integritní omezení ihned v relačním modelu.




**Obrázek 3.1.** Vytvoření relačního modelu v ERDPlus

[1] https://erdplus.com/

Po uložení diagramu je možné přejít do seznamu dokumentů a vygenerovat SQL příkazy pro vytvoření databáze. Nástroj dovoluje také zkopírovat výsledek, který je vidět na obrázku 3.2.

```
CREATE TABLE Person
(
    firstname NODATATYPE NOT NULL,
    surname NODATATYPE NOT NULL,
    dateOfBirth NODATATYPE NOT NULL,
    residence NODATATYPE NOT NULL,
    gender NODATATYPE NOT NULL,
    email NODATATYPE NOT NULL,
    PRIMARY KEY (firstname, surname, dateOfBirth, residence)
);

CREATE TABLE Employee
(
    position NODATATYPE NOT NULL,
    company NODATATYPE NOT NULL,
    firstname NODATATYPE NOT NULL,
    surname NODATATYPE NOT NULL,
    dateOfBirth NODATATYPE NOT NULL,
    residence NODATATYPE NOT NULL,
    PRIMARY KEY (firstname, surname, dateOfBirth, residence),
    FOREIGN KEY (firstname, surname, dateOfBirth, residence)
        REFERENCES Person(firstname, surname, dateOfBirth, residence)
);
```

**Obrázek 3.2.** Automaticky generované SQL dotazy v ERDPlus

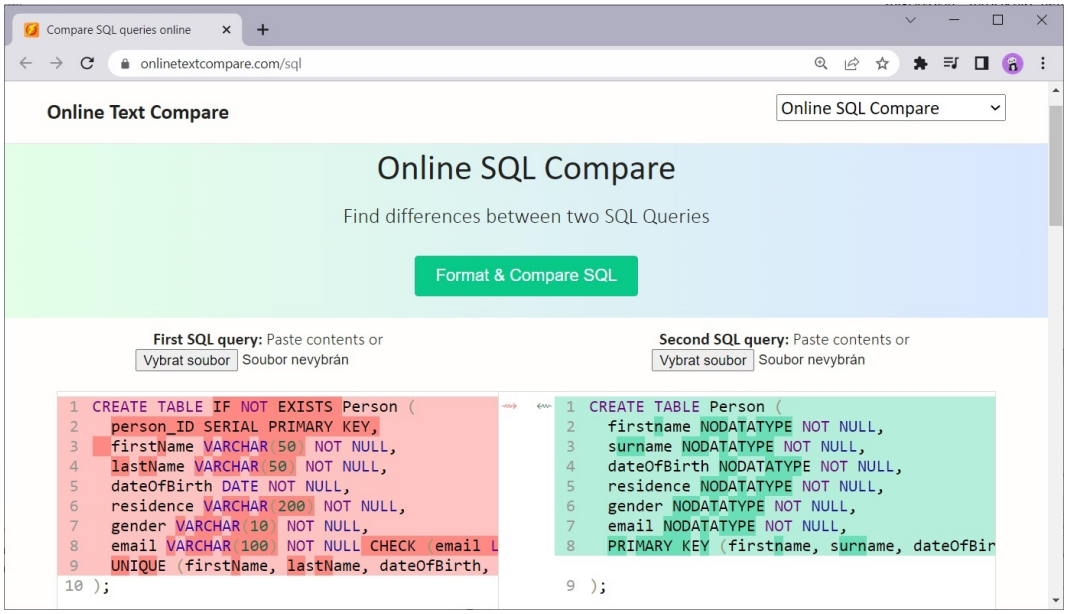
Jak je vidět, aplikace upřednostňuje použití tabulkových omezení, nepoužívá direktivy `ON DELETE/UPDATE` u cizích klíčů a nezavádí automaticky umělé klíče pro pohodlnou identifikaci tabulek při dotazování. Hlavní nevýhodou je, že studenti mají relační model v textové podobě, zatímco ERDPlus vyžaduje diagram. Jinak funkcionalita tohoto nástroje by mohla být dostatečná k generování SQL příkazů pro porovnání s řešeními studentů.

## 3.2 Porovnání SQL dotazů

Webová stránka Online Text Compare[2] dovoluje porovnávat SQL dotazy. Výhodou tohoto nástroje je možnost formátování dvou kódů, aby měli stejný tvar. Nástroj to pak textově porovná a označí rozdíly. Příklad použití pro dotaz vytvářející tabulku pro osobu ze sekce 1.1.2 a dotaz vygenerovaný v sekci 3.1 pomocí ERDPLus je na obrázku 3.3.

Nevýhodou nástroje je, že nerozlišuje prohozené řádky a nerozpozná shodné názvy tabulek nebo atributů, například `příjmení`, `surname` a `lastName`. Navíc není možné parametry porovnání ručně nastavovat, například není potřeba porovnávat typy atributů, protože z relačního modelu to není zřejmé a nebude se to kontrolovat v hodnotícím nástroji.

[2] `https://onlinetextcompare.com/sql`




**Obrázek 3.3.** Porovnání dvou SQL dotazů na webových stránkach Online Text Compare

## 3.3 Shrnutí existujících řešení

Oba zkoumané nástroje nejsou svou funkcionalitou jedinečné. Nevýhody uvedené v sekcích 3.1 a 3.2 jsou běžné a je těžké se jim vyhnout. Například je obtížné najít nástroj pracující s relačními modely v textové podobě. Vzhledem ke zmíněným v sekcích 3.1 a 3.2 nedostatkům stojí za to zvážit implementaci obdobné funkcionality s vlastními požadavky, jako je kupříkladu porovnávání atributů v tabulkách bez ohledu na jejich typy, rozpoznávání změněných názvů a používání příkazu `ALTER TABLE` pro přidání integritních omezení.

Dalším možným řešením problému porovnání schématu v databázi s relačním schématem je získat z databáze názvy vytvořených tabulek, jejich atributy, primární a cizí klíče a na jejich základě vygenerovat nový relační model. Srovnaní dvou relačních modelů by teoreticky mohlo být jednodušší než porovnání SQL dotazů.

# Kapitola 4
# Požadavky na nástroj

Důležitým krokem analýzy před implementací je určení cílové skupiny nástroje a požadavků, které bude systém splňovat. To pomůže nastínit rozsah projektu a strukturovat očekávání od nástroje.

## 4.1 Cílová skupina

K cílové skupině hodnoticího nástroje patří učitelé a studenti, kteří mají vztah k předmětu Databázové systémy. Používání nástroje bude následující:

- **učitelé** budou moci pomocí aplikace posoudit splnění základních požadavků zadání, které není potřeba ručně kontrolovat, a získat orientační počet bodů, který je vhodné studentovi udělit,
- **studenti**, kteří dělají úlohu tykající se převodu relačního modelu do SQL tabulek, budou moci pomocí nástroje zkontrolovat, zda splnili požadavky zadání, a získat nápovědu, co je potřeba přidat do svého řešení.

## 4.2 Business požadavky

Očekávání cílové skupiny od nástroje jsou definovány pomocí business požadavků. Každý z nich má vlastní kód, který ho jednoznačně identifikuje. Pro formulaci požadavků je použitá šablona doporučená v učebnici [7]: Jako `[role]` potřebuji `[cíl/přání]`, protože `[přínos]`.

- **[BRQ-1] Jako** student **potřebuji** rychlou průběžnou zpětnou vazbu během plnění úlohy, **protože** se chci ujistit, že se držím správného směru a že jsem nic nepřehlédl kvůli své nepozornosti.
- **[BRQ-2] Jako** učitel **potřebuji** automatickou kontrolu toho, zda student používá všechny požadované příkazy a operace, **protože** mohu něco přehlédnout a také **protože** chci věnovat více času a pozornosti logické stránce práce.
- **[BRQ-3] Jako** učitel **potřebuji** znát další názor na to, kolik bodů dát studentovi, **protože** chci objektivně hodnotit.
- **[BRQ-4] Jako** student **potřebuji** znát přibližný počet bodů, který můžu získat za své řešení, **protože** si chci rozhodnout, zda to už stačí na požadovanou známku za předmět.
- **[BRQ-5] Jako** uživatel **potřebuji** intuitivní ovládání nástroje, **protože** mi to usnadní jeho používání.

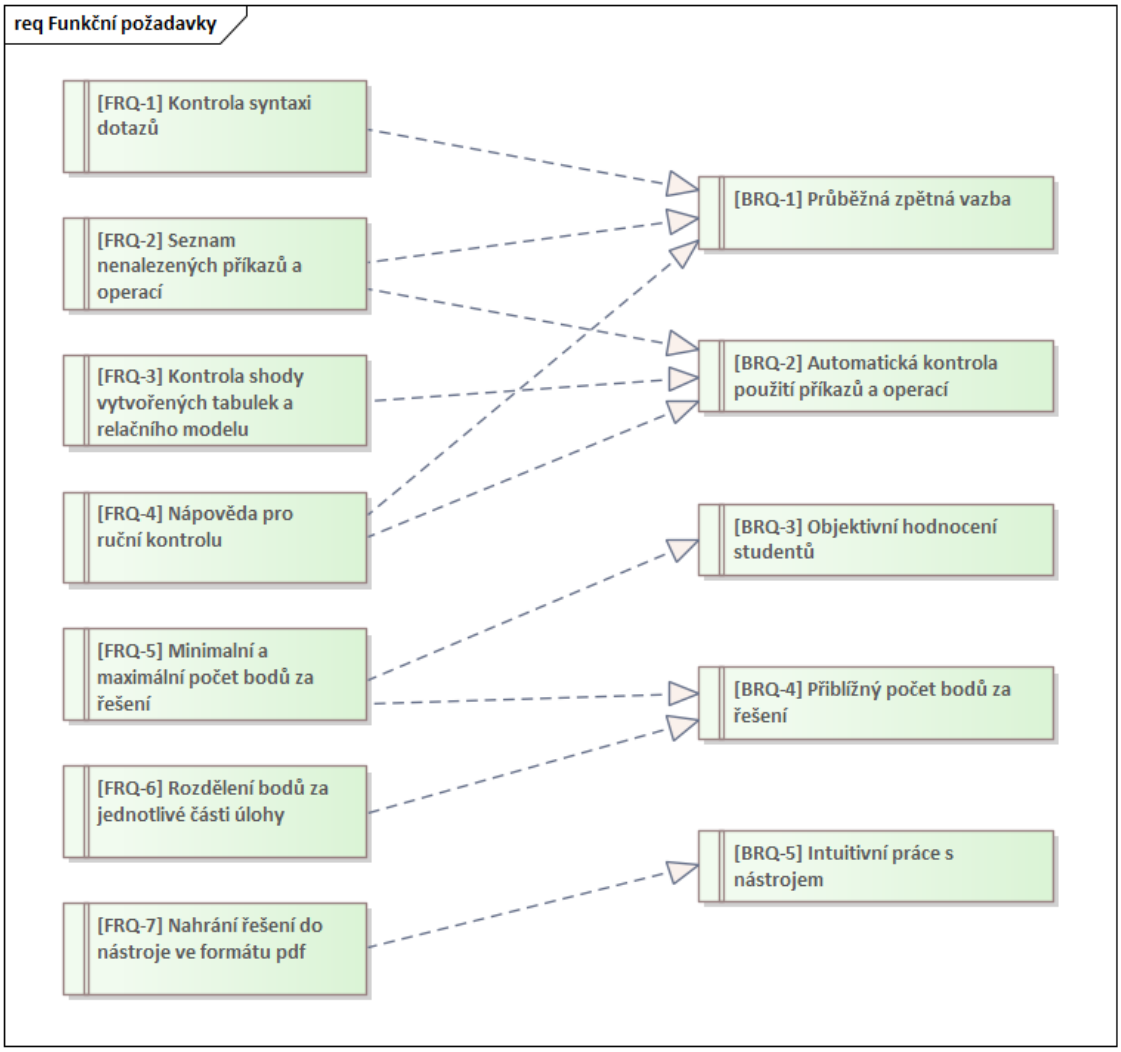
Psaní business požadavků je klíčové pro úspěšnou realizaci projektu, protože pomáhá definovat cíle a pochopit potřeby cílové skupiny. Tento proces také umožňuje vytvořit jasnou a srozumitelnou specifikaci projektu a lépe plánovat časový harmonogram pro dosažení stanovených cílů [8].

## 4.3 Funkční požadavky

Funkční požadavky popisují, jak budou business požadavky implementovány v systému, a uvádějí funkce dostupné uživatelům. Pro formulaci požadavků je použitá šablona doporučená v učebnici [7]: Systém umožní `[komu?]` `[co?]`.

- **[FRQ-1]** Systém umožní uživateli zkontrolovat syntaxi napsaných dotazů.
- **[FRQ-2]** Systém umožní uživateli zobrazit seznam nenalezených příkazů a operací.
- **[FRQ-3]** Systém umožní uživateli zjistit, zda vytvořené tabulky odpovídají relačnímu modelu.
- **[FRQ-4]** Systém umožní uživateli získat nápovědu, co má ručně zkontrolovat.
- **[FRQ-5]** Systém umožní uživateli zobrazit minimální a maximální počet bodů, který lze získat za aktuálně nahrané řešení.
- **[FRQ-6]** Systém umožní uživateli seznámit se s rozdělením bodů za jednotlivé části úlohy.
- **[FRQ-7]** Systém umožní studentovi nahrát řešení v požadovaném formátu pdf.

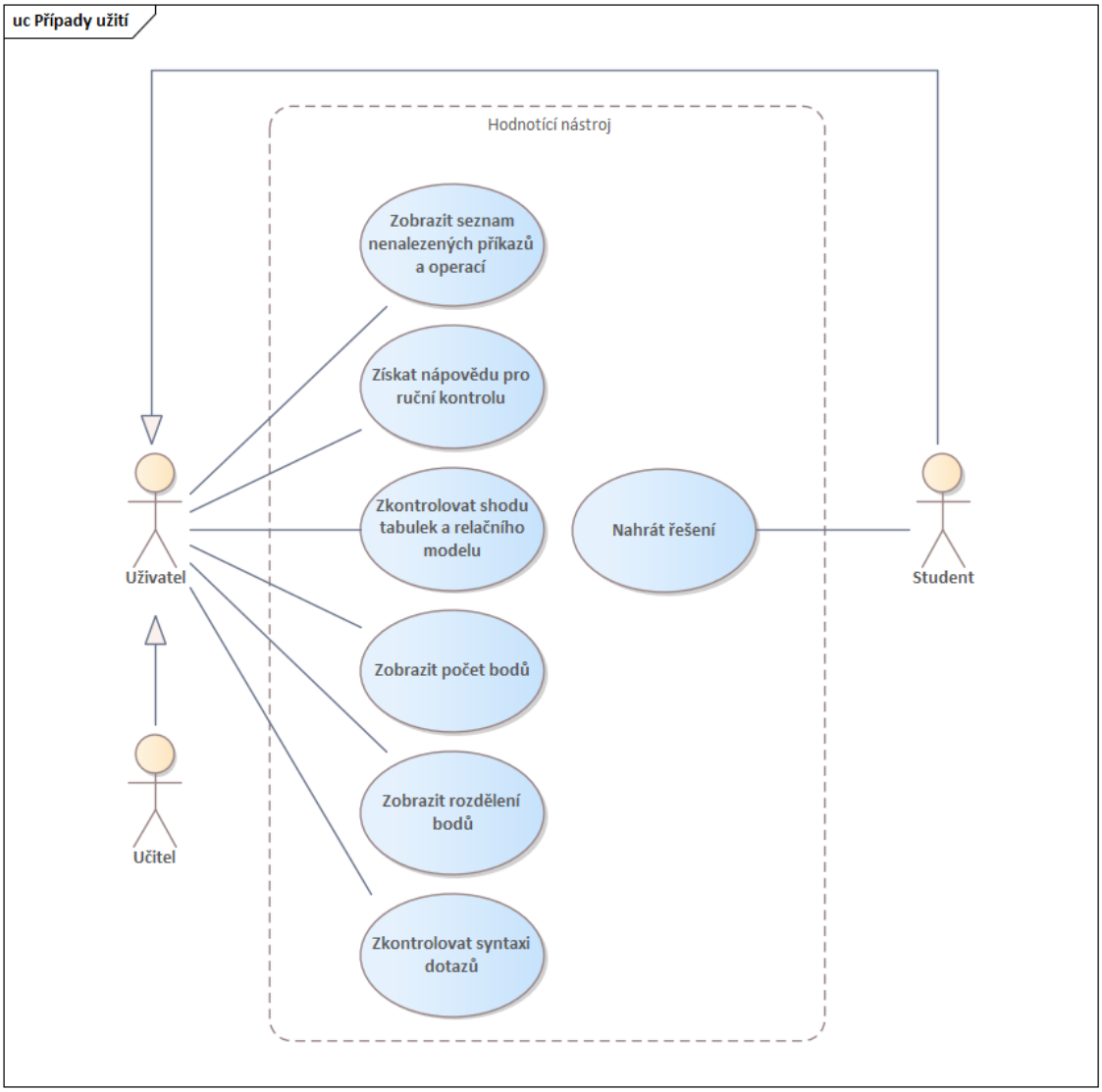
Propojení funkčních a business požadavků je znázorněno na obrázku 4.1.




**Obrázek 4.1.** Propojení funkčních a business požadavků

## 4.4 Případy užití

Diagram na obrázku 4.2 znázorňuje použití nástroje cílovou skupinou a uvádí způsoby interakce uživatelů s ním. Jak je vidět, způsoby používání těsně souvisí se systémovými požadavky. Všechny funkce, kromě nahrání řešení do nástroje, budou k dispozici všem uživatelům, tj. učitelům i studentům.




**Obrázek 4.2.** Diagram případů užití

# Kapitola 5
# Návrh

Po analýze samotné úlohy a existujících nástrojů, které by mohli pomoci při hodnocení studentských řešení, je třeba rozhodnout o architektuře hodnotícího nástroje a procesech, které v něm budou probíhat. To bude poslední krok před implementací.

## 5.1 Programovací jazyk

Pro implementaci hodnotícího nástroje jsou nejvhodnějšími programovacími jazyky Python nebo Java, které mají potřebné knihovny pro jednoduchou práci s PDF soubory, včetně jejich parsování. Oba jazyky jsou objektově orientované a platformně nezávislé.

V tomto projektu bude zvolen jazyk Java, pro který je dostupna knihovna Apache PDFBox[1]. Tato knihovna umožňuje uložit PDF soubor do proměnné, přečíst jeho obsah a v případě potřeby vygenerovat nový opravený PDF soubor. Navíc kritériem pro výběr jazyka Java je možnost jednoduchého průběžného testování pomocí jednotkových testů. Bude využita Java verze 11, aby nástroj byl kompatibilní se systémem BRUTE[2] pro odevzdání domácích úkolu v rámci Fakulty elektrotechnické ČVUT v Praze.

Pro případnou potřebu kontroly správnosti syntaxe SQL dotazů napsaných studenty stojí za zvážení také použití databáze, která by mohla provádět dotazy a po vyhodnocení řešení zanikat. Rozšíření Spring Frameworku v jazyce Java, které se nazývá Spring Boot, poskytuje rozsáhlou podporu pro práci s databázemi SQL, například umožňuje připojit vestavěnou do aplikace databáze H2, která bude spouštěna z téhož JVM (Java Virtual Machine), ve kterém běží program [9]. Pro komunikaci s tabulkovými zdroji dat se využívá přímý přístup JDBC (Java Database Connectivity) pomocí šablony JDBCTemplate, který je známý svou rychlostí a jednoduchostí použití v kódu [10].

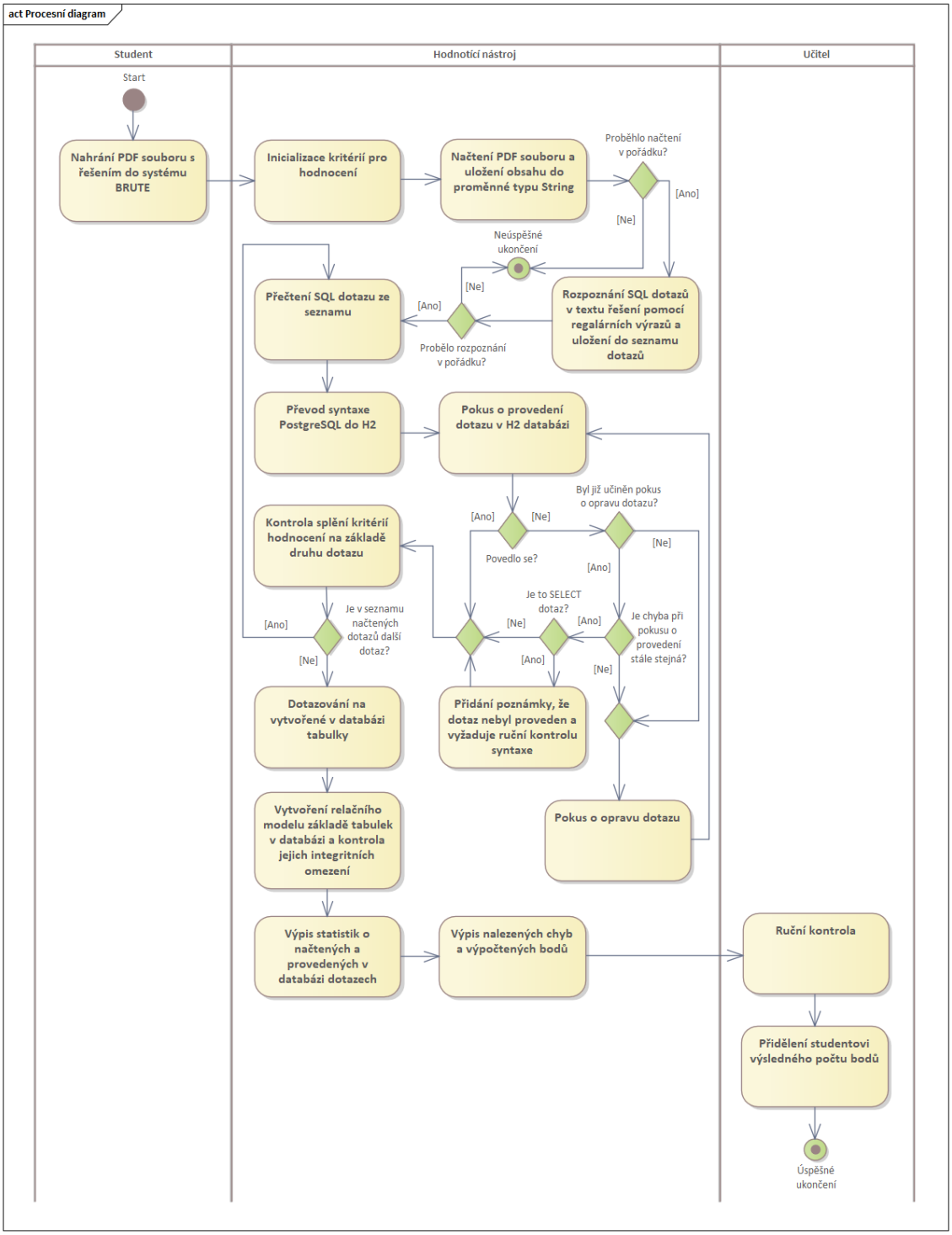
Kromě toho Spring Boot umožňuje vytvořit zdroje dat, které se automaticky použijí při spuštění programu. K vytvoření takového zdroje a připojení databáze stačí přidat závislosti a uvést konfigurační detaily [11]. V případě hodnotícího nástroje může to být užitečné při psání jednotkových testů.

## 5.2 Průběh procesu vyhodnocení

Při návrhu procesu vyhodnocení se bude předpokládat, že se automatická kontrola shody s relačním modelem zatím implementovat nebude. Vstupem do nástroje bude PDF soubor obsahující řešení zadání s SQL příkazy. Důležitým předpokladem pro fungování nástroje je, že studentovo řešení musí být v textové podobě, nikoli jako snímky obrazovky, jinak nebude možné obsah souboru parsovat. Ve fázi analýzy práce se zkontroluje syntaxe za použitím dotazování na databázi a následně se zkontroluje plnění požadavků na obsah dotazů. Na konci této fáze bude na základě tabulek vytvořených

[1] `https://mvnrepository.com/artifact/org.apache.pdfbox/pdfbox`
[2] `https://cw.felk.cvut.cz/brute/`

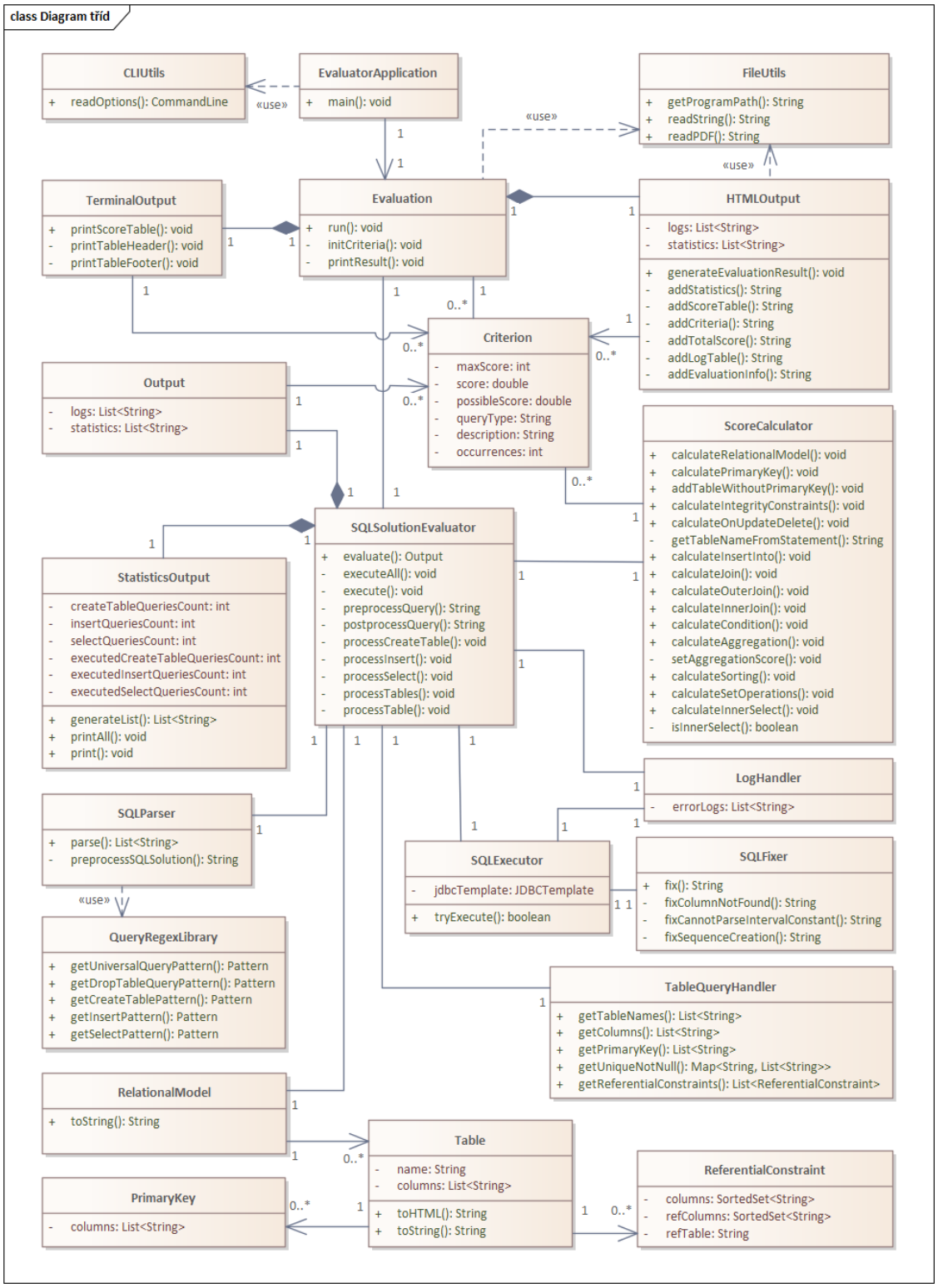
v databázi vygenerován relační model, který však bude třeba ručně porovnat s modelem studenta. Nakonec se vypíše statistika úspěšnosti práce nástroje s dotazy, tabulka s přidělenými body a vysvětlení výsledků hodnocení. Celý proces použití nástroje od nahrání PDF souboru studentem až do finálního hodnocení učitelem je znázorněn na obrázku 5.1.




**Obrázek 5.1.** Proces použití nástroje

## 5.3 Architektura

Architektura hodnotícího nástroje, průběh fungování kterého je popsán v sekci 5.2, je znázorněna na obrázku 5.2 jako diagram tříd.




**Obrázek 5.2.** Diagram tříd

Třída EvaluatorApplication inicializuje běh programu načtením parametrů potřebných ke spuštění z příkazového řádku. Volá funkci run třídy Evaluation, která je zodpovědná za celý proces vyhodnocování. Jejím hlavním nástrojem je SQLSolutionEvaluator, který kompletně analyzuje dotazy v řešení studenta, načtené třídou SQLParser pomocí regulárních výrazů. Provádí tyto dotazy v databázi prostřednictvím nástroje SQLExecutor s případnou opravou chyb za použitím třídy SQLFixer a průběžně ten proces loguje v třídě LogHadler. Po dokončení všech dotazů se SQLSolutionEvaluator ptá databáze na vytvořené tabulky pomocí metod třídy TableQueryHandler a generuje relační model obsahující seznam tabulek s jejich primárními a cizími klíči. Ke zhodnocení splnění kritérií používá třída SQLSolutionEvaluator třídu ScoreCalculator. Na konci běhu programu se zobrazí výstup v příkazovém řádku a ve formátu HTML stránky, která obsahuje různé statistiky a výsledek vyhodnocení.

# Kapitola 6
# Implementace

Proces implementace se řídil návrhem z kapitoly 5. Všechny funkční požadavky uvedené v sekci 4.3 byly implementovány s výjimkou automatické kontroly shody s relačním modelem. Tento požadavek je částečně realizován tak, že relační model je generován na základě tabulek vytvořených v databázi, ale podobnost s modelem studenta je třeba kontrolovat ručně.

## 6.1 Inicializace

Běh programu se začíná načtením čtyř parametrů: cesty k řešení s SQL dotazy, cesty k relačnímu modelu, cesty k souboru s kritérii hodnocení a názvu výstupního souboru s výsledky hodnocení. Druhý a čtvrtý parametry nejsou povinné. Cesta k relačnímu modelu není potřeba proto, že ten se nijak nezpracovává z důvodu omezené funkčnosti hodnotícího nástroje. Poslední parametr je zodpovědný pouze za název výstupního souboru. Když není uveden, bude se soubor nazývat „output“ podle výchozího nastavení. Hodnotící nástroj lze spustit za použitím následující šablony příkazu:

```
java -jar evaluator.jar                          \
     -sql solution/fileWithQueries.pdf           \
     -rm  solution/fileWithRelationalModel.pdf   \
     -c   fileWithCriteria.json                  \
     -o   outputFileName.html
```

Jak již bylo zmíněno v sekci 5.1, studentovo řešení se načítá za použitím knihovny Apache PDFBox, výstupem je obsah PDF souboru s SQL dotazy ve formátu String, který je v Javě určen pro ukládání a práci s posloupnostmi symbolů, neboli textem [12]. Během implementace se objevilo, že knihovna neumí zpracovávat záhlaví a zápatí stránek, takže se občas dostávají do studentových dotazů, což vede k chybám při jejich automatické kontrole.

Kritéria hodnocení jsou uložena v JSON souboru pro případ, že by bylo třeba změnit rozdělení bodů za jednotlivé části zadání. Stojí za to upozornit, že pokud budou změněny názvy kritérií nebo jejich počet, nástroj nemusí pracovat správně. Formát JSON byl zvolen pro konfiguraci kritérií, protože pomocí knihovny Jackson je možné jednoduše převést obsah souboru na Java objekty [13].

## 6.2 Parsování

Soubor s řešením, který student odevzdává do systému BRUTE, obsahuje kromě SQL dotazů také doplňující text s poznámkami pro učitele. Aby bylo možné toto řešení softwarově zpracovat, je třeba poznámky odstranit a jednotlivé dotazy vyjmout pro následující automatickou kontrolu.

Zřejmým by se mohlo zdát použití existujícího parseru. Například, jsou v Javě open source knihovny JSqlParser[1] a JOOQ[2] určené pro rozpoznání standardních SQL dotazů s jednoduchou syntaxí, avšak pokročilé syntaktické konstrukce nejsou v nich definované. Vzhledem ke specifičnosti úlohy pro automatickou kontrolu, která předpokládá použití konstrukcí, jako je vnořený `SELECT`, nebylo možné žádnou z těchto knihoven použít.

Z tohoto důvodu bylo rozhodnuto vyhledávat dotazy pomocí ručně vytvořených regulárních výrazů sloužících podle definice k efektivnímu zpracování textu a vyhledávání specifických vzorů v něm [14]. Ale napsat regulární výraz, který by rozpoznal jakýkoli dotaz, téměř není možné, protože gramatika jazyka SQL je příliš složitá a různorodá, má hodně variant syntaxe a podporuje mnoho datových typů, výrazů, funkcí a příkazů, což ještě více ztěžuje tento úkol. Proto bylo třeba zanedbat definici syntaxe uvnitř dotazu, čímž se stanovily dva hlavní požadavky na regulární výrazy v rámci hodnotícího nástroje:

- *určení začátku dotazu.* Tento problém byl zpracován tak, že byly sestaveny vlastní regulární výrazy pro několik typů dotazů: `CREATE TABLE`, `DROP TABLE`, `INSERT INTO`, `SELECT`, které jsou nezbytnou součástí studentova řešení, a univerzální regulární výraz pro další často používané dotazy jako `CREATE SEQUENCE`, `ALTER TABLE`, `DELETE FROM`. Regulární výrazy vyhledávají podle klíčových slov, kterými začíná jakýkoli dotaz určitého typu v jazyce SQL.
- *rozpoznání konce dotazu.* Řešení tohoto problému se ukázalo jako nemožné bez definování syntaxe celého dotazu, ale jak bylo napsáno dříve v této sekci, bylo rozhodnuto toto zanedbat. Jediným způsobem, jak definovat konec dotazu, je tedy středník.

## 6.3 Vyhodnocení

Hodnocení probíhá podle kriterií a rozložení bodů rozepsaných v sekcích 2.1 a 2.4. Ze seznamu kriterií, které by teoreticky bylo možné automaticky kontrolovat, se nepovedlo kvůli časovému omezení implementovat porovnání studentova relačního modelu a relačního modelu generovaného z vytvořených v databázi tabulek. Tím pádem může student získat maximálně 18 bodů po kontrole hodnotícím nástrojem. Nicméně program generuje relační model založený na syntakticky správných `CREATE TABLE` dotazech, což učiteli umožňuje pohodlně provést srovnání ručně.

Samotný proces vyhodnocení probíhá tak, že nejdřív program zkouší provést dotaz ve H2 databázi vestavěné do aplikace. Protože studenti píší SQL dotazy v dialektu PostgreSQL [15], který se mírně liší od dialektu H2 [16], databáze někdy nemůže dotazy provést. V takových případech může program některé zjevné chyby opravit a vyzkoušet provedení dotazu znovu. Pokud byl dotaz proveden v databázi, je považován za syntakticky správný a postupuje do další fáze kontroly, která je určena typem dotazu:

- Ihned po syntaktické validaci se v `CREATE TABLE` dotazech hledá použití integritních omezení, s výjimkou primárních klíčů, a direktiv `ON UPDATE` a `ON DELETE` podle klíčových slov. Nejobtížnější část kontroly však přichází po provedení všech dotazů. Po vytvoření tabulek v databázi se program ptá na jejich názvy, primární a cizí klíče. V tomto okamžiku program konečně zkontroluje použití primárních klíčů a na základě obdržených údajů pak vygeneruje relační model pro porovnání.
- Dotazy typu `INSERT INTO` jsou kontrolovány pouze na přítomnost těchto dotazů v studentově řešení a na jejich syntaktickou správnost.

[1] `https://jsqlparser.sourceforge.net/`
[2] `https://www.jooq.org/`

- Dotazy typu `SELECT` jsou jednoduše kontrolovány podle kritérií pomocí vyhledávání klíčových slov. Jejich zvláštností je, že se kontrolují, i když nebyly provedeny v databázi. Důvodem je, že problém určení konce dotazu není zcela vyřešen. Pokud student neumístí středník na konce řady dotazů, může se stát, že bude načteno několik dotazů najednou. Databáze tak nebude schopna správně spustit dotazy, i když mohou být syntakticky správné. Kromě toho, databáze také nebude schopna provést dotazy, pokud v databázi se kvůli syntaktickým chybám nevytvoří potřebné tabulky. Takové dotazy jsou programem označeny jako „nespolehlivé“ a později na výstupu jsou označeny jinak než dotazy provedené.

## 6.4 Výstup hodnotícího nástroje

Výstupem hodnotícího nástroje je HTML soubor, obsahující statistiku načtení a provedení v databázi jednotlivých `CREATE TABLE`, `INSERT INTO` a `SELECT` dotazů, seznam chyb během dotazování na H2 databázi, tabulku s body a podrobnější zdůvodnění výsledků vyhodnocení.

Příklad bodové tabulky je znázorněn na obrázku 6.1. Ve sloupcích zleva doprava jsou uvedeny popisy kritérií, maximální počet bodů za splnění každého z nich, počet získaných bodů, teoretický počet bodů, pokud jsou zkontrolované, ale neprovedené dotazy správné, typy dotazů, které jsou kontrolovány na splnění kritéria, a počet toho, kolikrát bylo každé kritérium splněno.

| DESCRIPTION | MAX SCORE | CERTAIN SCORE | POSSIBLE SCORE | QUERY TYPE | OCCURRENCES |
|---|---|---|---|---|---|
| defining primary keys of tables | 1 | 1.0 | 1.0 | CREATE | 13 |
| using integrity constraints | 2 | 2.0 | 2.0 | CREATE | 13 |
| using ON UPDATE/DELETE for foreign key | 2 | 2.0 | 2.0 | CREATE | 10 |
| statements to fill tables by data | 2 | 0.0 | 0.0 | INSERT | 0 |
| outer join of tables | 2 | 0.0 | 2.0 | SELECT | 1 |
| inner join of tables | 2 | 0.0 | 2.0 | SELECT | 2 |
| condition on the data | 1 | 1.0 | 1.0 | SELECT | 3 |
| aggregation | 2 | 2.0 | 2.0 | SELECT | 1 |
| sorting and pagination | 1 | 1.0 | 1.0 | SELECT | 2 |
| set operations | 1 | 0.0 | 1.0 | SELECT | 1 |
| inner SELECT | 2 | 0.0 | 2.0 | SELECT | 2 |
| **TOTAL SCORE** | **18** | **9.0** | **16.0** | | |

**Obrázek 6.1.** Bodová tabulka na výstupu hodnotícího nástroje

Nejzajímavější ze seznamu kritérií je „defining primary keys of tables“, pro které se body počítají z poměru tabulek s primárním klíčem vytvořených pomocí `CREATE TABLE` dotazů úspěšně provedených v databázi ke všem přečteným dotazům tohoto typu.

Tabulka s body, statistika načtení a provedení dotazů a seznam chyb ve formě logů se zobrazí také na příkazovém řádku. Logování je realizováno pomocí Java knihovny Log4j[3]. Logování je užitečným nástrojem pro diagnostiku chování programu za jeho běhu. Díky uchovávání záznamů o průběhu programu je možné v pozdější fázi identifikovat a řešit odhalené problémy [17].

[3] `https://logging.apache.org/log4j/2.x/`

# Kapitola 7
# Testování funkcionality

Po implementaci byl hodnotící nástroj pokryt jednotkovými testy za použitím Java knihoven JUnit[1] a Mockito[2]. Jednotkové testování slouží k ověření, zda veškerý kód vyhovuje stanoveným standardům kvality před jeho nasazením, zjednodušuje proces ladění a při případném dalším vývoji projektu může navíc pomoci zorientovat se v již existujících funkcích [18].

Poté byl hodnotící nástroj otestován na skutečných řešeních studentů předmětu Databázové systémy. Bylo použito 40 řešení z letního semestru akademického roku 2021/2022 a 41 řešení z letního semestru akademického roku 2022/2023.

## 7.1 Analýza řešení z akademického roku 2021/2022

Předpokládalo se, že fáze testování řešení z akademického roku 2021/2022 bude těsně spojená s finálním laděním programu a že během tohoto období bude odhalena a odstraněna převážná část nedostatků v programu.

Jako jediná chyba bránící programu v jeho funkci se objevilo to, že se během parsování mohlo dojít k přetečení zásobníku. Tato chyba byla způsobena omezením výchozí knihovny pro regulární výrazy v jazyce Java. Pro opravu této chyby byla provedena úprava, kdy byla délka dotazu získaného z regulárního výrazu omezena na přibližně 1400 znaků v závislosti na dotazu.

Program měl další významný nedostatek spočívající v absenci některých funkcí, které studenti používali ve svých dotazech v dialektu PostgreSQL, a které nebyly k dispozici v H2. To mohlo vést k chybám při provádění těchto dotazů. Pro řešení tohoto problému lze využít příkaz `CREATE ALIAS`, který umožňuje přidat vlastní funkce do databázového schématu, přičemž lze je psát v jazyce Java, prostřednictvím obklopení kódu funkce dvojitými znaménky dolaru. Tento příkaz je možné spustit pomocí konfiguračního souboru, který je automaticky vykonáván při každém spuštění programu Spring Boot. Tímto způsobem budou funkce, které byly manuálně definovány v PostgreSQL, dostupné i v H2. Jako příklad může sloužit přidání chybějící funkce `split_part` do H2, což je znázorněno na obrázku 7.1.

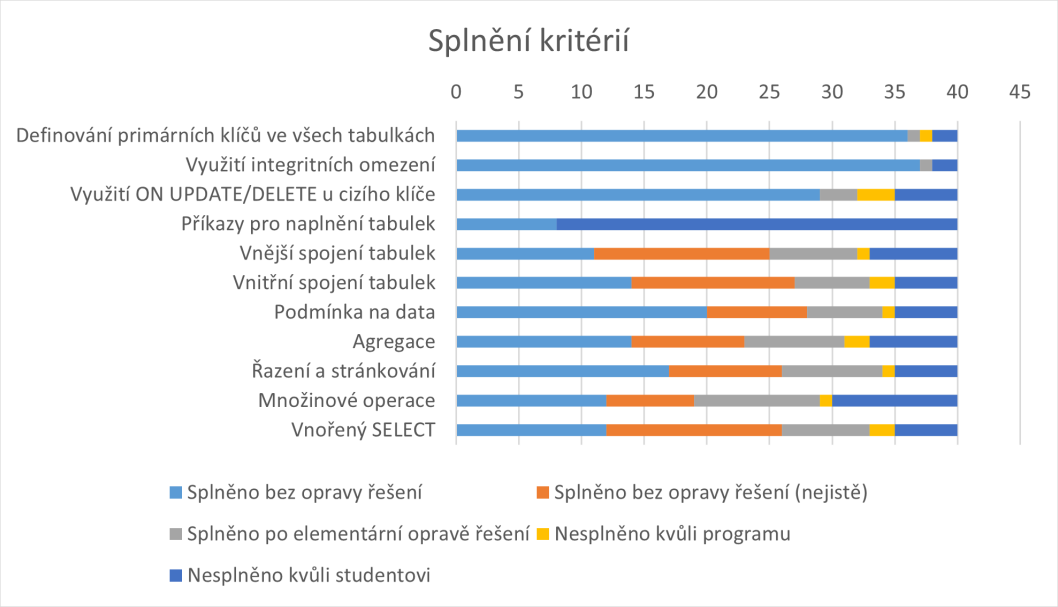
```
DROP ALIAS IF EXISTS SPLIT_PART;
CREATE ALIAS SPLIT_PART AS '
import java.text.*;
@CODE
String SPLIT_PART(String originalString, String delimiter, int index) throws Exception {
    return originalString.split(delimiter)[index-1];
}
';
```

**Obrázek 7.1.** Přidání funkce z PostgreSQL do H2

[1] https://junit.org/
[2] https://site.mockito.org/

Navíc byla vyřešena řada menších chyb, které omezovaly kontrolu dotazů. Především se tyto opravy týkaly transformací syntaxe PostgreSQL na H2 před a po zkoušení provedení dotazů. Většinu chyb s tím spojených se povedlo různými způsoby upravit. H2 databáze ale nezná klíčová slova `FULL OUTER JOIN` a tento problém se vyřešit nepodařilo.

Na obrázku 7.2 je statistika úspěšnosti nástroje při analýze řešení z akademického roku 2021/2022.




**Obrázek 7.2.** Statistika vyhodnocení řešení z akademického roku 2021/2022

Na začátku byly všechny práce podrobeny kontrole pomocí hodnotícího nástroje v jejich původní podobě bez jakýchkoli úprav. Při této kontrole bylo zohledněno, zda lze jednoznačně potvrdit splnění kritérií úlohy (na obrázku 7.2 označeno světle modře), nebo zda je třeba provést ruční ověření (na obrázku 7.2 označeno oranžově). Ve druhém případě se jednalo o `SELECT` dotazy, kde se při pokusu o jejich provedení v databázi vyskytly chyby. Pokud se v práci vyskytly elementární chyby, které bránily úplnému otestování řešení, byly tyto chyby opraveny a práce byla následně znovu zkontrolována nástrojem (na obrázku 7.2 označeno šedě). Mezi tyto chyby patří například překlepy, použití klíčových slov syntaxe SQL jako názvy tabulek nebo atributů, použití stejného názvu pro různá integritní omezení nebo chybějící středníky na koncích dotazů.

Po provedení kontroly může být kritérium označeno jako nesplněné ve dvou případech: buď kritérium nebylo ve skutečnosti splněno studentem (na obrázku 7.2 označeno tmavě modře), nebo jeho splnění nebylo rozpoznáno nástrojem kvůli přehlédnutému nedostatku v programu (na obrázku 7.2 označeno žlutě).

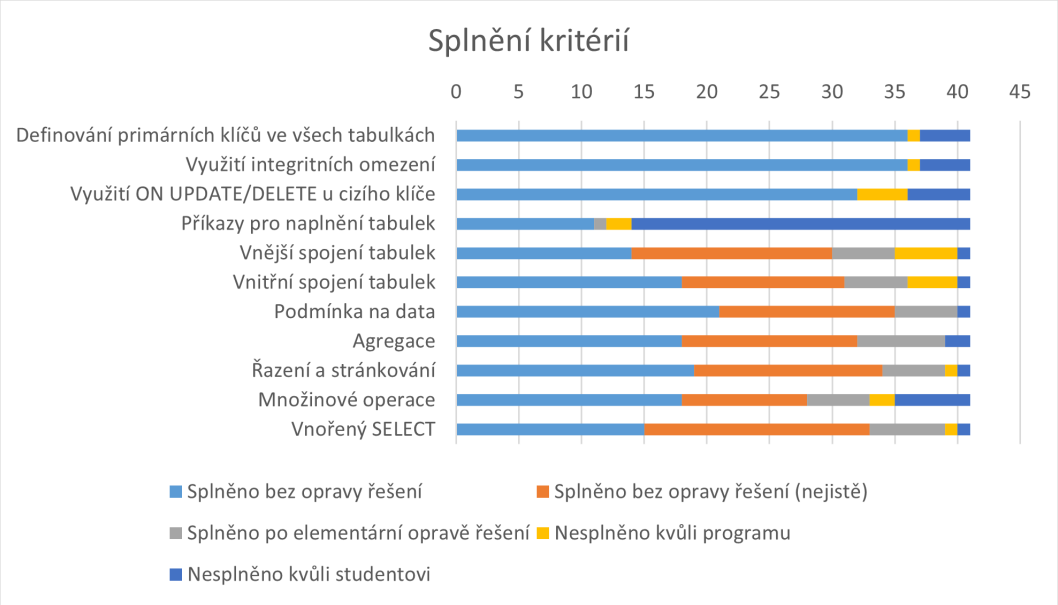
## 7.2 Analýza řešení z akademického roku 2022/2023

Po fázi ladění byl program otestován na nových řešeních z akademického roku 2022/2023, aby bylo možné porovnat úspěšnost s předchozím rokem a posoudit použitelnost hodnotícího nástroje.

V této fázi došlo k rozhodnutí změnit vyhodnocení použití agregace v `SELECT` dotazu. Agregace zahrnuje použití agregačních funkcí, jako je `AVG`, `SUM`, `COUNT`, a klíčových slov `GROUP BY` a `HAVING`. Dříve byl udělován plný počet bodů za splnění kritéria alespoň

jedním z těchto způsobů. Nově je však udělován 1 bod za použití alespoň jedné agregační funkce a další 1 bod za použití `GROUP BY` a `HAVING`. Maximální počet bodů za splnění tohoto kritéria je tedy 2.

Statistika úspěšnosti nástroje při analýze řešení z akademického roku 2021/2022 je znázorněna na obrázku 7.3. V porovnání s předchozím rokem došlo ke zvýšení počtu kritérií, která byla automaticky vyhodnocena jako splněná, bez zásahu do studentova řešení. Celkově v obou letech přesahuje množství úspěšně nalezených splnění kritérií 75 % při zohlednění případů pokusů o opravu chyb.




**Obrázek 7.3.** Statistika vyhodnocení řešení z akademického roku 2022/2023

## 7.3 Shrnutí

Po provedení testování byl hodnotící nástroj konečně integrován do systému BRUTE, zatím je však k dispozici pouze jako pomůcka pro učitele a není přístupný pro studenty. Pokud by studenti měli možnost opravit elementární chyby ve svých řešeních, bylo by možné úspěšnost při hledání splnění kritérií hodnotit na více než 75 %. Většina případů, kdy bylo kritérium označeno programem jako nesplněné, nebyla způsobena nedostatky v programu, ale tím, že dané kritérium nebylo pokryto v řešení. Výsledek je vzhledem k stanoveným cílům uspokojivý, i když program zatím nelze bezpečně používat bez další ruční kontroly.

Nejčastějším důvodem, proč nelze zkontrolovat, zda je řešení zcela správné, je to, že mnoho řešení nemá na koncích dotazů středníky. Kontrole také brání omezení související s rozdíly v syntaxi PostgreSQL a H2. Dalšími častými chybami, kterých se studenti dopouštěli, bylo například používání klíčových slov z jazyka SQL jako názvů tabulek nebo atributů a používání stejných názvů pro různá integritní omezení.

# Kapitola 8
# Závěr

Cílem projektu bylo seznámit se s úlohou zadávanou v předmětu Databázové systémy, analyzovat možnosti její automatické evaluace a navrhnout řešení problému automatického vyhodnocování. Poté bylo nutné navržený hodnotící nástroj implementovat a otestovat ho kontrolou skutečných příkladů řešení úlohy.

K dosažení tohoto cíle byl nejprve proveden podrobný rozbor zadání v kapitole 2, včetně rozdělení bodů za jednotlivé části a zkoumání chyb, které se v řešeních studentů vyskytují. Chyby byly rozděleny do kategorií a označeny kódy pro snadnou orientaci, viz tabulka 2.3. Bylo rozhodnuto, že některé chyby, například logické, je obtížné kontrolovat automaticky, a proto to nebude od nástroje vyžadováno.

Poté byla provedena analýza webových nástrojů pro převod relačního modelu do SQL dotazů a pro textové porovnání SQL dotazů, viz kapitola 3. Vzhledem k specifičnosti úlohy v předmětu Databázové systémy a z toho plynoucích možných problémů v procesu vyhodnocení bylo rozhodnuto nevyužít existující řešení a odložit implementaci překladu mezi relačním modelem a SQL dotazy.

Výsledkem analýzy je stanovení cílové skupiny a funkčních požadavků nástroje, popis procesu používání a návrh architektury s ohledem na výběr programovacího jazyka Java. To všechno, včetně diagramů, je popsáno v kapitolách 4 a 5.

Hodnotící nástroj byl realizován v souladu s navrhovaným designem a procesy popsané v kapitole 6 byly pokryty jednotkovými testy. Po implementaci byla funkčnost programu otestována na 81 studentských řešeních. Výsledky testů lze dohledat v kapitole 7. Podle nich nástroj odhalí splnění kritérií v pracích studentů v 75 % případů. Většina případů, kdy program označil kritérium jako nesplněné, nebyla způsobena nedostatky v programu, ale tím, že dané kritérium nebylo skutečně splněno v řešení.

## 8.1 Další vývoj projektu

Vzhledem k tomu, že program zatím není schopen fungovat autonomně a automaticky kontrolovat řešení studentů, projekt předpokládá další vývoj. Nejprve by mělo být implementováno porovnání relačního modelu vygenerovaného z databázových tabulek s relačním modelem studenta, což zvýší celkový počet bodů při automatické analýze prací na 23. Pak by bylo nutné řešit rozdíl mezi syntaxí PostgreSQL vyučovanou v předmětu Databázové systémy a používanou studenty a syntaxí H2 databáze využitou v hodnotícím nástroji.

Vývoj projektu by mohl být zaměřen i na vylepšení schopnosti rozpoznávání dotazů v textu, zejména při čtení dotazů bez středníku na konci. Pro dosažení tohoto cíle by bylo vhodné zvážit pokročilejší přístup k práci s regulárními výrazy a případně i použit nějakým způsobem hotový parser. Kromě toho, v Javě existují nástroje určené pro rozpoznávání jazyků, například ANTLR[1], které poskytují široké možnosti parsování a analýzy, což dovoluje získávat informace o struktuře dotazů a provádět s nimi další operace. Tento přístup by teoreticky mohl přispět k efektivnímu řešení problému.

[1] `https://www.antlr.org/`

# Literatura


[1] Tomáš Řehák. *Inovace a automatizace procesů*. 2011, cit. 2023-01-17. `https://is.muni.cz/th/qorqb/thesis.pdf`. Supervisor: doc. RNDr. Tomáš Pitner, Ph.D.

[2] Michal Valenta Jaroslav Pokorný. *Databázové systémy*. 1. vyd.. Praha: České vysoké učení technické v Praze, 2013. ISBN 978-80-01-05212-9.

[3] Martin Svoboda. *Přednášky z předmětu Databázové systémy*. 2019, cit. 2022-10-18. `https://www.ksi.mff.cuni.cz/~svoboda/courses/182-B0B36DBS/`.

[4] Zbyněk Bureš. *Databázové systémy 1*. 1. vyd.. Jihlava: Vysoká škola polytechnická Jihlava, 2014. ISBN 978-80-87035-88-7.

[5] Radim Farana. *Tvorba relačních databázových systémů*. 2. vyd.. Ostrava: Vysoká škola báňská Technická univerzita, 2004. ISBN 80-7078-706-6.

[6] Graeme C. Simsion a Graham C.Witt. *Data Modeling Essentials*. 3. vyd.. San Francisco: Elsevier Inc., 2005. ISBN 0-12-644551-6.

[7] Miroslav Bureš a kol. *Efektivní testování softwaru : klíčové otázky pro efektivitu testovacího procesu*. 1. vyd.. Praha: Grada Publishing, 2016. ISBN 978-80-247-5594-6.

[8] Robin F. Goldsmith. *Discovering Real Business Requirements for Software Project Successu*. Artech House, 2004. ISBN 978-15-805-3771-1.

[9] *H2 Database Web, Features*. cit. 2023-05-12. `http://www.h2database.com/html/features.html`.

[10] Donald Bales. *JDBC pocket reference*. O'Reilly Media, Incorporated, 2003. ISBN 978-05-960-0457-6.

[11] Dinesh Rajput. *Designing Applications with Spring Boot 2.2 and React JS*. BPB Publications, 2019. ISBN 978-93-893-2805-9.

[12] Rogers Cadenhead. *Sams Teach Yourself Java in 24 Hours*. Sams, 2012. ISBN 978-06-723-3575-4.

[13] Jeff Friesen. *Java XML and JSON*. Apress, 2019. ISBN 978-14-842-4330-5.

[14] Jeffrey E. F. Friedl. *Mastering regular expressions*. 3. vyd.. Sebastopol: O'Reilly, 2006. ISBN 978-05-965-2812-6.

[15] *PostgreSQL Database Web*. cit. 2023-05-13. `https://www.postgresql.org/docs/current/`.

[16] *H2 Database Web, Grammar*. cit. 2023-05-13. `http://www.h2database.com/html/grammar.html`.

[17] Samudra Gupta. *Logging in Java with the JDK 1.4 Logging API and Apache Log4j*. 2. vyd.. Apress, 2014. ISBN 978-14-302-5398-3.

[18] Dave Thomas Jeff Langr, Andy Hunt. *Pragmatic Unit Testing in Java 8 with JUnit*. Pragmatic Bookshelf, 2015. ISBN 978-16-805-0424-8.

# Příloha A
# Struktura externích příloh

| | |
|---|---|
| **./evaluator** | Projekt v jazyce Java 11 s hodnotícím nástrojem, který je implementován pomocí Maven[2]. |
| **./evaluator-jar** | Složka obsahuje soubory potřebné pro spuštění hodnotícího nástroje a jeho integraci do systému BRUTE. |
| `evaluator.jar` | Java projekt zabalený do souboru typu JAR a připravený ke spuštění. |
| `criteria-map.json` | Soubor, ze kterého nástroj načítá kritéria pro bodování. |
| `solution/sql.pdf` | Příklad řešení úlohy s SQL dotazy potřebnými ke spuštění nástroje. |
| `solution/rm.pdf` | Příklad řešení úlohy s relačním modelem, který může být v budoucnu použit pro porovnání. |
| **./example** | Složka obsahuje příklad použití hodnotícího nástroje. |
| `sql.pdf` | Ukázka řešení úlohy s SQL dotazy. |
| `output.html` | Ukázka výstupu nástroje po provedení automatické evaluace. |
| **./test-statistics** | Složka obsahuje výsledky testování studentských řešení. |
| `statistics-2022.xlsx` | Statistika splnění kritérií při kontrole řešení z akademického roku 2021/2022. |
| `statistics-2023.xlsx` | Statistika splnění kritérií při kontrole řešení z akademického roku 2022/2023. |

[2] `https://maven.apache.org/`