Špecifikácia projektu

Katedra softwarového inženýrství

Matematicko-fyzikální fakulta, Univerzita Karlova

Riešiteľ: Bc. Richard Hvizdoš

Študijny program: Informatika - Softwarové a datové inženýrství

Názov projektu: Základ frameworku pre efektívne profilovanie multi-modelových dát

Typ projektu: Výzkumný projekt

Vedúci: Ing. Pavel Koupil, Ph.D.

Konzultant: doc. RNDr. Irena Holubová, Ph.D., Mgr. Jáchym Bártík

1 Úvod

V rámci výskumnej skupiny pod vedením docentky I. Holubovej[1] je momentálne vyvíjaný systém na samoadaptívnu správu multi-modelových dát. Tento systém je navrhnutý tak, aby minimalizoval užívateľskú interakciu a rozhodovacie procesy v zložitých scenároch. Avšak efektívna aplikácia metód strojového učenia vyžaduje významné množstvo reálnych multi-modelových dát, ktoré sú ideálne bez anomálií a odľahlých hodnôt, čím sa zabezpečí čistota dát.

V súčasnosti existuje množstvo nástrojov na profilovanie dát, ktoré sú založené na detekcii funkčných závislostí [1]. Napriek tomu sú tieto nástroje primárne prispôsobené pre relačné dáta (napr. vo formáte CSV) a ich schopnosť efektívne spracovávať veľké dátové súbory je obmedzená, čo negatívne vplýva na ich škálovateľnosť.

Cieľom tohto projektu je nadviazať na projekty výskumnej skupiny a navrhnúť nástroj, ktorý využije distribuované rámce na efektívnejšiu detekciu funkčných závislostí vo veľkých multi-modelových dátových súboroch.

1.1 Vzťah k ostatným projektom

Výskumná skupina už vyvinula niekoľko nástrojov na správu multi-modelových dát, vrátane MM-infer [2] na odvodenie multi-modelových schém, MM-cat [3] na konceptuálne modelovanie multi-modelových dát, MM-evocat [4] na propagáciu zmien schémy do dátových inštancií a MM-quecat [5] na dotazovanie nad konceptuálnym modelom multi-modelových dát. Navrhovaný nový nástroj má za cieľ túto sadu rozšíriť o modul na profilovanie dát. Tento nástroj bude nadväzovať na schopnosti MM-infer pri extrakcii multi-modelových schém a využije distribuované frameworky na efektívnejšiu detekciu funkčných závislostí vo veľkých multi-modelových dátových súboroch.

V budúcnosti sa očakáva, že tento modul bude slúžiť ako základ pre komplexnejší nástroj zameraný na efektívne profilovanie rozsiahlych multi-modelových dát. Konkrétne, zlepší extrakciu funkčných závislostí začlenením pravých funkčných závislostí, inkluzívnych závislostí a odmietacích obmedzení. Ďalej bude podporovať extrakciu integritných obmedzení a komplexných obchodných pravidiel, detekciu odľahlých hodnôt, identifikáciu redundancií v rôznych dátových súboroch a implementáciu modulu na porovnávanie schém. Toto porovnávanie schém sa nebude spoliehať len na ontológiu, ale aj na zarovnanie pravých funkčných závislostí.

[1] `https://www.ksi.mff.cuni.cz/area.html?id=multi-model-data`

V rámci tohto projektu však vznikne iba jadro tohto veľkého nástroja. Jadro sa zameriava na efektívne profilovanie veľkých multi-modelových dát a zároveň umožní efektívne využívanie metód strojového učenia na vývoj ďalších nástrojov určených na samoadaptívnu správu multi-modelových dát.

2 Popis projektu

Cieľom tohto projektu je vytvoriť nástroj obsahujúci algoritmy na hľadanie funkčných závislostí v rôznych populárnych formátoch (napríklad CSV, JSON), so zameraním na BIG DATA. Z tohto dôvodu je požadovaná vysoká škálovateľnosť, rýchlosť a presnosť.

Známe existujúce prístupy na hľadanie funkčných závislostí sa obvykle stretávajú s problémom podpory iba jedného formátu na vstupe, problémom so škálovateľnosťou, optimalizáciou algoritmu na počet atribútov alebo n-tíc vstupu a taktiež s nedostatočným testovaním na BIG DATA[2] (napríklad paper porovnávajúci 7 rôznych algoritmov [1]). Medzi tieto algoritmy patria napríklad TANE [6], FDEP [7], FastFDs [8], Dep-Miner [9]. Odbočením od týchto klasických algoritmov je hybridný prístup kombinujúci riadkovo a stĺpcovo optimálne riešenie HyFD [10] alebo aproximácia funkčných závislostí algoritmom Aid-FD [11] (tento prístup aproximácie nebude braný do úvahy).

Inšpiráciou pre algoritmy je ich implementácia od Hasso-Plattner-Institut v jazyku Java[3], ktorú čiastočne prevezmem, následne upravím a paralelizujem do frameworku Apache Spark. Ako základná inšpirácia modulu na spracovanie dát poslúži ich projekt Metanome[4] ponúkajúci pridanie rôznych druhov algoritmov na spracovanie dát.

Momentálne projekt cieli na akademické prostredie, a preto napríklad SW časť nástroja nebude obsahovať všetky potrebné časti, ktoré sú požadované pri komerčnom využití a distribúcii nástroja, napríklad nebude implementovať používateľov a používateľské roly.

Výstupy tohto projektu tvoria základ diplomovej práce, ktorá bude nadväzovať na tento projekt pochopením algoritmov, zistením ich slabých stránok a navrhnutím algoritmu na hľadanie funkčných závislostí v BIG DATA, ktorý by tieto nedostatky eliminoval. Ďalej je možné pracovať napríklad na detekcii redundancií alebo hľadaní integritných obmedzení dát a vhodnom rozširovaní vytvoreného modulu.

2.1 Funkčné požiadavky

Algoritmus bude vo vstupných dátach vyhľadávať funkčné závislosti na základe užívateľom zadaných parametrov:

- formát súboru výberom z možností (napr. CSV, JSON),
- vstupný súbor, v ktorom sa budú vyhľadávať funkčné závislosti,
- upresnenie, či vstupný súbor obsahuje hlavičku, v prípade súborov formátu CSV (v tých je hlavička voliteľná a popisuje názvy atribútov),
- špecifikácia oddeľovača hodnôt v súbore formátu CSV (oddeľovače sú typicky "," alebo ";"),

[2] datasety s veľmi veľkým počtom atribútov alebo riadkov

[3] `https://hpi.de/naumann/projects/repeatability/data-profiling/fds.html`

[4] `https://hpi.de/naumann/projects/data-profiling-and-analytics/metanome-data-profiling.html`

- limit na počet spracovaných záznamov zo vstupného súboru,
- počet záznamov, ktoré budú preskočené vo vstupnom súbore pred začiatkom algoritmu,
- limit na veľkosť LHS funkčnej závislosti, ktorým sa môže predísť nájdeniu náhodných funkčných závislostí,
- určenie výstupu algoritmu (výstup na konzolu alebo do súboru).

Nástroj na prácu s dátami a algoritmom musí byť spoľahlivý, s možnosťou spustenia algoritmu a nastavenia parametrov, ktoré budú nastavené výberom z ponuky alebo napísaním hodnoty do textového políčka. Po vyhodnotení algoritmu bude zobrazený výsledok a štatistiky ako napríklad doba, po ktorú algoritmus bežal, počet nájdených funkčných závislostí a ich samotné vypísanie, nameraná spotreba pamäti alebo informovanie užívateľa o progrese algoritmu pri jeho vyhodnocovaní.

2.2 Ne-Funkčné požiadavky

Okrem funkčných požiadaviek je potrebné splniť aj nasledujúce ne-funkčné požiadavky, ktoré charakterizujú kvalitu nástroja:

- Vysoký výkon algoritmov pri hľadaní funkčných závislostí.
- Stabilný modul a algoritmy s vhodným výpisom chybových hlášok.
- Jednoduchá inštalácia a spustenie, podrobná používateľská a programátorská dokumentácia.
- Modulárna architektúra s rozdelením nástroja na frontend a backend.

2.3 Workflow

Algoritmus bude pracovať na obvyklom princípe: načíta vstupný dataset, pomocou daného prístupu bude vyhľadávať funkčné závislosti, ktoré nakoniec vydá. Nutnú rýchlosť dosiahneme aj použitím distribuovaného systému Apache Spark na vhodných častiach algoritmu, ktoré nebudú ovplyvnené distribúciou výpočtu.

Samotné spustenie algoritmu a jeho potrebné nastavenie bude vykonávať užívateľ vo vyvíjanom nástroji, ktorý komunikuje s frontendom. Práca s nástrojom bude vo webovom prehliadači, keďže nástroj bude vo forme webovej stránky.

Ako prvé bude potrebné uložiť dataset do samotného nástroja s možnosťou určenia vlastného názvu alebo popisu (pre vlastnú potrebu). Formát súboru predvyplní SW, ale užívateľ by ho mal skontrolovať.

Algoritmy na hľadanie funkčný závislostí budú v nástroji uložené. Užívateľ následne môže vyhľadávať funkčné závislosti zvolením algoritmu, datasetu s nastavením potrebných parametrov a zvolením výstupu. Údaje budú v správny čas overované (na strane klienta aj servera) a chybové hlášky budú zobrazované užívateľovi. Frontend potom odošle požiadavku na backend, kde nastane spracovanie a spustenie algoritmu. Počas jeho behu bude frontend a backend komunikovať, aby sa informácie o progese zobrazovali užívateľovi v nástroji.

Po dokončení algoritmu sa nájdené funkčné závislosti zobrazia v nástroji a uložia na určenom mieste so štatistikami o behu algoritmu. Tie bude možné zobrazovať v záložke nástroja.

2.4 Mockups

Na nasledujúcich obrázkoch je možné vidieť návrh, ako by mohol samotný nástroj vyzerať s približným uložením plánovaných funkcionalít. Prostredie bude jednoduché na obsluhu a orientáciu s jazykom anglickým.

Na obrázku 1 je zobrazená úvodná obrazovka, ktorá sa otvorí pri spustení nástroja. Z menu je možné vyberať zo všetkých dostupných funkcií, ktoré nástroj ponúka. Prvým krokom bude uloženie datasetu do nástroja (na obrázku 2) na jeho následné spracovanie.

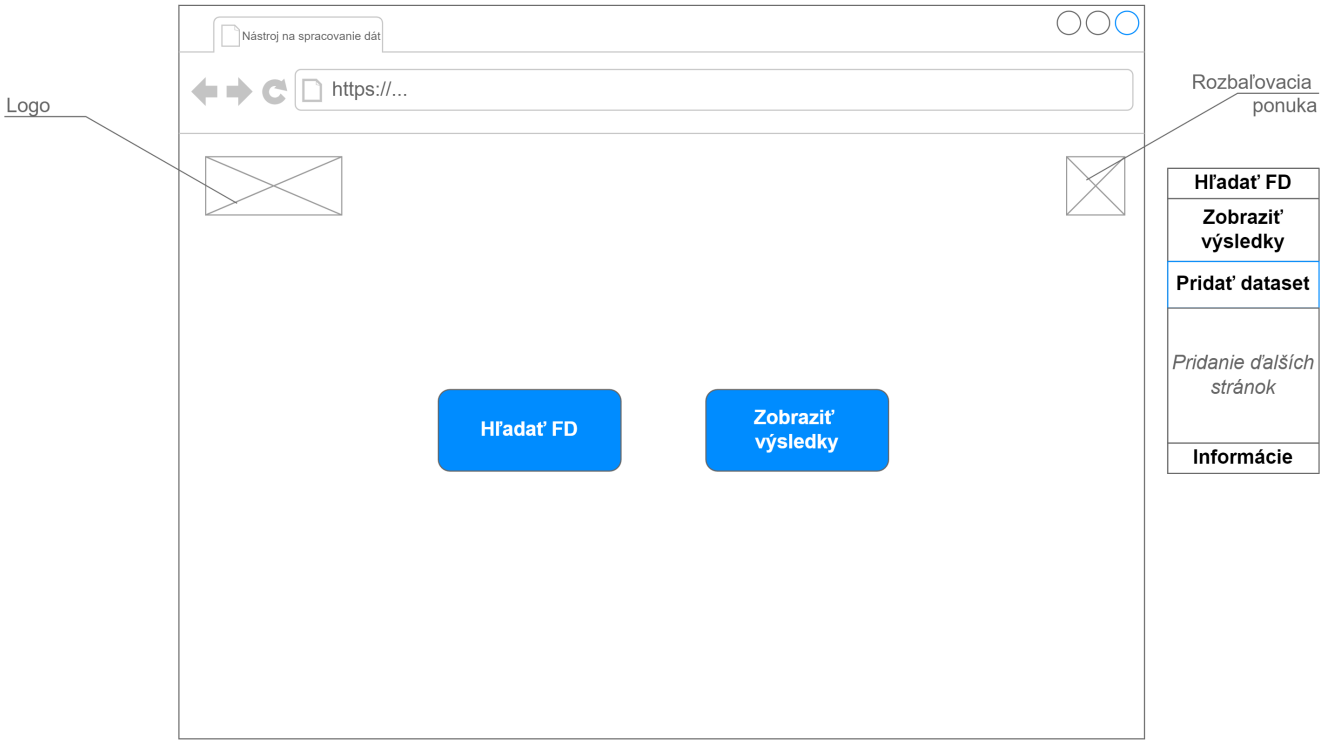

Figure 1: Návrh dizajnu úvodnej obrazovky nástroja.

Spracovanie datasetu bude prebiehať na stránke Vyhľadať FD, zobrazenej na obrázku 3, s nastavením všetkých potrebných parametrov. Popri vykonávaní sa bude zobrazovať progres daného výpočtu a užívateľ bude dostatočne informovaný. Po dokončení hľadania bude možné si pozrieť výsledky so štatistikami na stránke zobrazenej na obrázku 4.

Na stránke Informácie, ktorá nie je zobrazená v mockups, bude základný text o projekte, kto na projekte pracoval a je zaň zodpovedný, a samozrejme aj dokumentácia (resp. odkaz na ňu).

2.5 Architektúra

Na obrázku 5 je možné vidieť jednoduchý návrh architektúry nástroja pozostávajúceho z frontendu (webovej aplikácie) a backendu napojeného na potrebné komponenty.

Každá z týchto častí bude vykonávať svoju štandardnú prácu. Frontend, s ktorým bude komunikovať používateľ, bude generovať webové stránky, predávať požiadavky a zobrazovať chybové hlášky na informovanie používateľa.

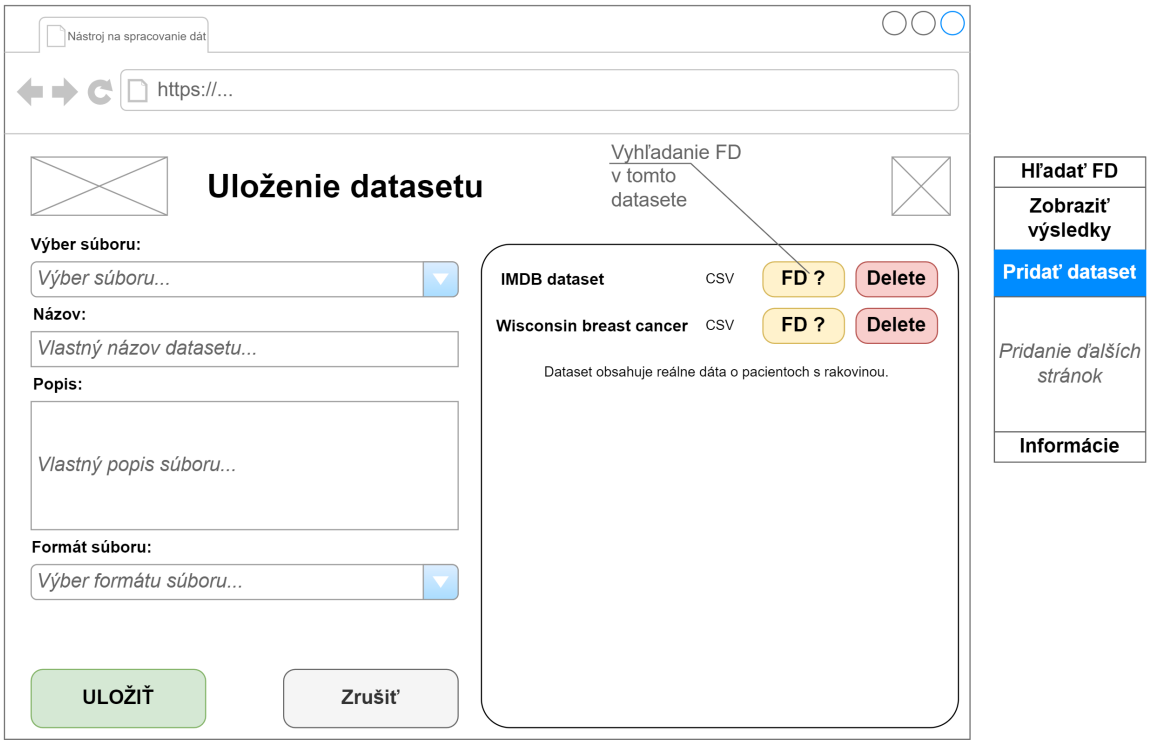

Figure 2: Návrh dizajnu obrazovky na uloženie datasetu nástroja.

Figure 3: Návrh dizajnu obrazovky na spustenie algoritmu hľadania FD nástroja.

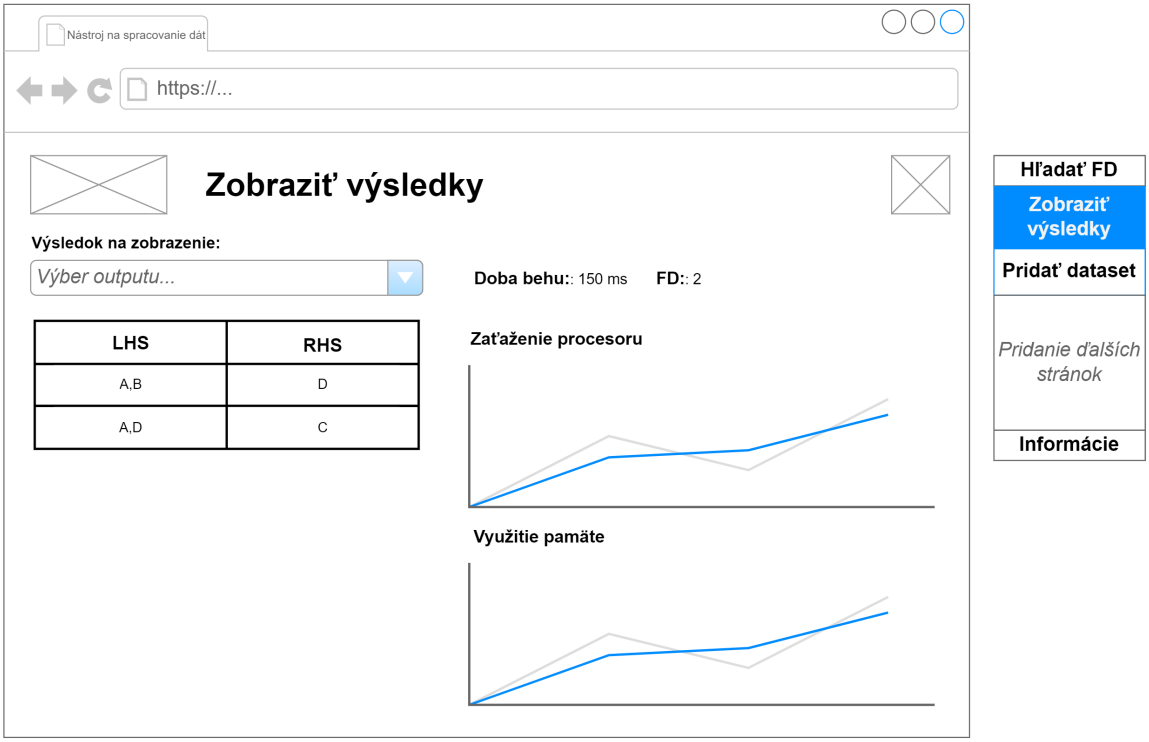

Figure 4: Návrh dizajnu obrazovky zobrazujúcej výsledky nástroja.

Backend, s ktorým sa bude komunikovať pomocou API, bude obsahovať business logiku, proces na spustenie algoritmu a jeho kontrolu, a následne predanie výsledkov.

Algoritmy na hľadanie funkčných závislostí budú spúšťané v Apache Spark, z dôvodu dosiahnutia rýchlosti a škálovateľnosti. Na zjednodušenie bude nástroj ukladať dáta lokálne v databáze alebo file systéme zariadenia. Do budúcna sa plánuje aj pripojenie na externú databázu prostredníctvom nového modulu "Načítanie dát z DB", s pripojením napríklad na Cassandru alebo Neo4j.

Celý nástroj bude vyvíjaný tak, aby v budúcnosti bolo možné jeho rozšírenie a využitie už dostupných častí.

3 Platformy a technológie

Algoritmus na hľadanie funkčných závislostí bude programovaný v Jave[5], nástroj: frontend v JavaScripte[6], HTML, s použitím nástroja ako napríklad React[7]; backend v Jave použitím Spring boot[8]. Na zrýchlenie algoritmu bude použitý aj distribuovaný systém Apache Spark[9].

[5] https://www.java.com/en/
[6] https://www.javascript.com/
[7] https://react.dev/
[8] https://spring.io/projects/spring-boot
[9] https://spark.apache.org/

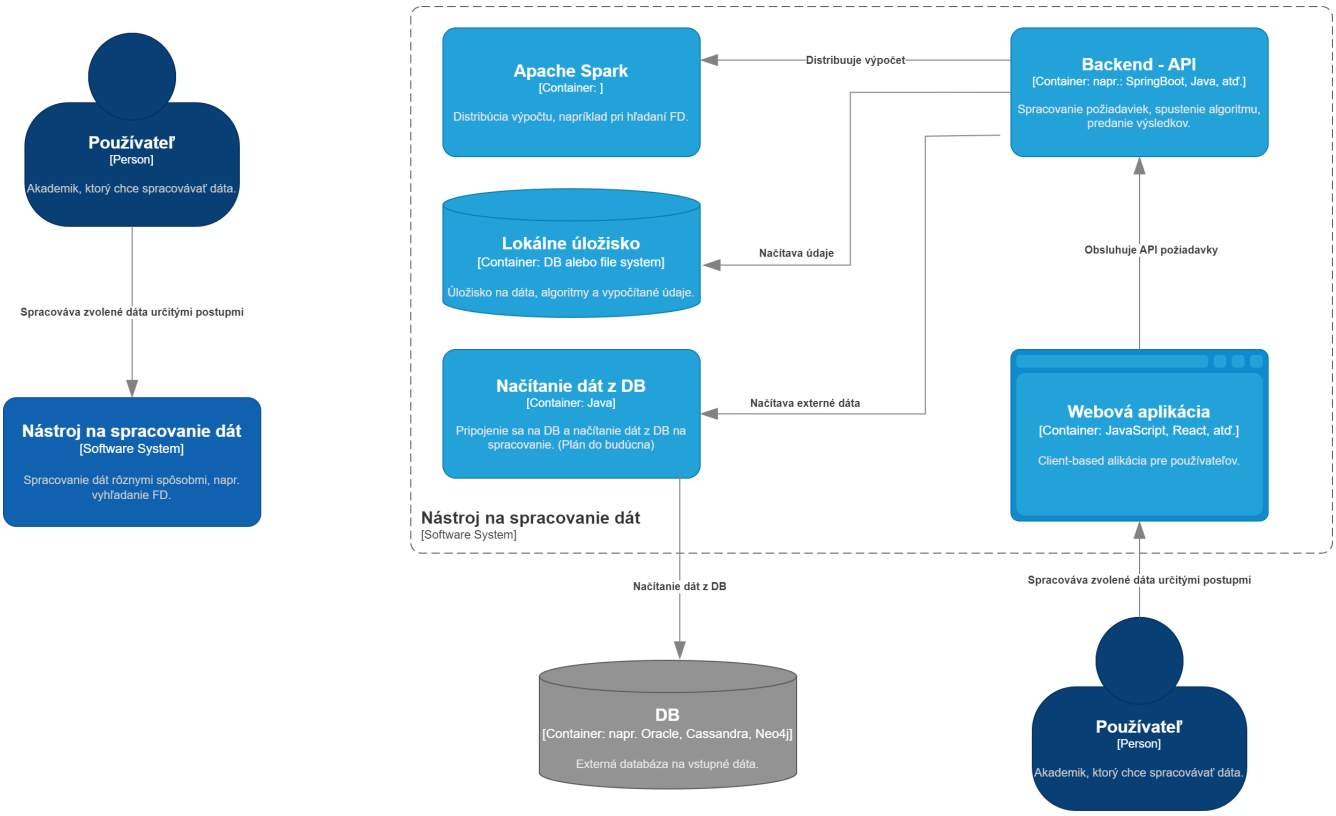

Figure 5: Ukážka L1 a L2 návrhu architektúry nástroja.

4 Risky

Risky, ktoré môžu nastať pri práci na tomto projekte sú napríklad:

- Vybrané prístupy nebude možné efektívne paralelizovať pomocou frameworku Apache Spark, napr. z dôvodu komplexných a na sebe závislých operáciách. Jedným z dôsledkov môže byť paralelná implementácia, ktorá ale bude menej efektívna ako pôvodná implementácia.
- Vlastnosti dátových formátov typických pre multi-modelové dáta môžu obsahovať prvky, ktoré nebude možné namapovať na vlastnosti relačných dát (napr. absenciu hodnoty v JSON dokumente je možné vyjadriť rôznymi spôsobmi, ktoré sú ale odlišné od spôsobov typických pre relačné dáta). Multi-modelová extenzia tak môže byť časovo náročná, a to z dôvodu vyžiadania si mapovania na relačný model - možné riešenie napr. nahradením relačného modelu ako jadra existujúcich prístupov za viac abstraktný model.
- Upravené a paralelizované prístupy nebudú dostatočne škálovateľné z dôvodu ich nevhodného výpočtu a hľadania funkčných závislostí. Možnosťou je použiť iba časti algoritmov, ktoré sú dostatočne škálovateľné a zvyšné časti upraviť.

5 Míľniky / Výstupy

Prvým míľnikom bude **Prototyp**, ktorý bude implementovať backend s implementovaným jedným algoritmom na hľadanie funkčných závislostí. Tento prototyp sa následne bude testovať na overenie základnej funkčnosti nástroja.

Prvá verzia projektu bude obsahovať viaceré algoritmy a implementovať celú architektúru vrátane frontendu. Nebude chýbať príprava nástroja na jeho budúcu integráciu do nástroja MM-cat.

Nasledovať bude testovanie nástroja a všetkých algoritmov, ich evaluácia a spracovanie podrobnej dokumentácie.

6 Časový harmonogram

Časový harmonogram je navrhnutý s ohľadom na skúsenosti riešiteľa, náročnosť projektu a vykonanú analýzu. V implementácii je zahrnutý čas 15 MD na implementovanie a úpravu algoritmov, zvyšný čas na vytvorenie nástroja. Čas na testovanie bude najviac pohltený samotným vytváraním testov, ich spúšťaním a riešením výsledkov.

Nezanedbateľnú časovú dotáciu má aj evaluácia nástroja a algoritmov, a taktiež vytvorenie kvalitnej dokumentácie, ktorá je podstatná pre ďalšie fungovanie a vývoj nástroja.

Míľnik / Výstup	**Aktivita**	**Časový plán**	**Man-days**
Prototyp	Implementácia	Mesiac 1 - 2	15
	Testovanie	Mesiac 2	2
Prvá verzia	Implementácia	Mesiac 3 - 4	20
	Nasadenie	Mesiac 4	2
	Evaluácia	Mesiac 4	6
Dokumentácia		Mesiac 4 - 5	5

7 Forma spolupráce

Na projekte budem spolupracovať s vedúcim práce, ako aj s ostatnými členmi tými, ktorí sa tejto problematike venujú.

Rešitel (Richard Hvizdoš) prevezme plnú zodpovednosť za návrh, implementáciu a testovanie novo vyvinutého nástroja na extrakciu funkčných závislostí z veľkých multi-modelových dátových súborov. Súčasne bude rešitel spolupracovať so zvyšnými členmi tímu, ktorí poskytnú informácie týkajúce sa rozhrania nástroja MM-cat, aby uľahčili integráciu modulu do väčšieho projektu.

7.1 Plán konzultácií

Konzultácie a synchronizačné stretnutia so všetkými členmi tímu sa budú konať raz za štyri týždne, každé s dĺžkou deväťdesiat minút. Okrem toho sa konzultácie s vedúcim projektu uskutočnia na týždennej báze po dobu šesťdesiat minút, s výnimkou týždňov, kedy sa stretáva celý tím.

7.2 Spolupráca s ostatnými členmi tímu

Všetci členovia tímu sa zúčastnia osobných konzultácií na Malostranskom námestí. Tieto konzultácie sa budú primárne zameriavať na diskusie týkajúce sa architektúry a rozhrania nástroja, ako aj na potenciálne optimalizácie s cieľom zabezpečiť jeho použiteľnosť v kontexte zostávajúcich aplikácií vyvíjaných tímom doc. I. Holubovej. Okrem toho sa konzultácie budú zaoberať integráciou knižníc implementovaných v rámci nástroja MM-cat, vrátane schematickej kategórie pre reprezentáciu schémy a wrapperov pre implementáciu systémovo špecifických vlastností základných systémov (konkrétne zdrojov dát, z ktorých sa extrahujú funkčné závislosti).

Konzultácie s vedúcim projektu sa budú primárne sústrediť na implementáciu konkrétnych algoritmov a ich kvalitatívne požiadavky. Tieto požiadavky zahŕňajú úplnosť a správnosť implementácie, škálovateľnosť, podporu rôznych dátových modelov a vývoj vlastného algoritmu, ktorý primerane zohľadňuje funkčné požiadavky a (protichodné) vlastnosti dnešných populárnych dátových modelov.

Literatúra

1. PAPENBROCK, Thorsten; EHRLICH, Jens; MARTEN, Jannik; NEUBERT, Tommy; RUDOLPH, Jan-Peer; SCHÖNBERG, Martin; ZWIENER, Jakob; NAUMANN, Felix. Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms. *Proc. VLDB Endow.* 2015, vol. 8, no. 10, pp. 1082–1093.
2. KOUPIL, Pavel; HRICKO, Sebastián; HOLUBOVÁ, Irena. A universal approach for multi-model schema inference. *J. Big Data*. 2022, vol. 9, no. 1, p. 97. Available from DOI: 10.1186/S40537-022-00645-9.
3. KOUPIL, Pavel; SVOBODA, Martin; HOLUBOVÁ, Irena. MM-cat: A Tool for Modeling and Transformation of Multi-Model Data using Category Theory. In: *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS 2021 Companion, Fukuoka, Japan, October 10-15, 2021*. IEEE, 2021, pp. 635–639. Available from DOI: 10.1109/MODELS-C53483.2021.00098.
4. KOUPIL, Pavel; BÁRTÍK, Jáchym; HOLUBOVÁ, Irena. *MM-evocat:* A Tool for Modelling and Evolution Management of Multi-Model Data. In: HASAN, Mohammad Al; XIONG, Li (eds.). *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*. ACM, 2022, pp. 4892–4896. Available from DOI: 10.1145/3511808.3557180.
5. KOUPIL, Pavel; CRHA, Daniel; HOLUBOVÁ, Irena. A universal approach for simplified redundancy-aware cross-model querying. *Inf. Syst.* 2025, vol. 127, p. 102456. Available from DOI: 10.1016/J.IS.2024.102456.
6. HUHTALA, Ykä; KÄRKKÄINEN, Juha; PORKKA, Pasi; TOIVONEN, Hannu. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *Comput. J.* 1999, vol. 42, pp. 100–111.
7. FLACH, Peter; SAVNIK, Iztok. Database Dependency Discovery: A Machine Learning Approach. *AI Commun.* 1999, vol. 12.
8. WYSS, Catharine; GIANNELLA, Chris; ROBERTSON, Edward. FastFDs: A Heuristic-Driven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances. 2001, vol. 2114.

9. LOPES, Stéphane; PETIT, Jean-Marc; LAKHAL, Lotfi. Efficient Discovery of Functional Dependencies and Armstrong Relations. 2000, vol. 1777, pp. 350–364. ISBN 978-3-540-67227-2. Available from DOI: 10.1007/3-540-46439-5_24.

10. PAPENBROCK, Thorsten; NAUMANN, Felix. A Hybrid Approach to Functional Dependency Discovery. 2016, pp. 821–833. Available from DOI: 10.1145/2882903.2915203.

11. BLEIFUSS, Tobias; BÜLOW, Susanne; FROHNHOFEN, Johannes; RISCH, Julian; WIESE, Georg; KRUSE, Sebastian; PAPENBROCK, Thorsten; NAUMANN, Felix. Approximate Discovery of Functional Dependencies for Large Datasets. 2016, pp. 1803–1812. Available from DOI: 10.1145/2983323.2983781.