FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

Možnosti analytického rozšíření úložiště Data Lakehouse

Josef Bozděch

PLZEŇ

2024

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

Možnosti analytického rozšíření úložiště Data Lakehouse

Bc. Josef Bozděch

Vedoucí práce
Doc. Dr. Ing. Jana Klečková

PLZEŇ 2024

Citace v seznamu literatury:
BOZDĚCH, Josef. `Možnosti analytického rozšíření úložiště Data Lakehouse` . Plzeň, 2024. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Doc. Dr. Ing. Jana Klečková.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Josef BOZDĚCH**
Osobní číslo:	**A22N0039P**
Studijní program:	**N0613A140040 Softwarové a informační systémy**
Téma práce:	**Možnosti analytického rozšíření úložiště Data Lakehouse**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s konceptem úložiště Data Lakehouse a jeho prototypem v rámci Medical Research and Education.
2. Seznamte se s dostupnými kolekcemi dat v úložišti Medical Research and Education a způsoby jejich využívání.
3. Zvolte a implementujte vhodné analytické a statistické metody dle požadavků kladených na Data Lakehouse.
4. Řešení uživatelsky a funkčně otestujte.
5. Vyhodnoťte získané výsledky a porovnejte s dosavadními výsledky systému Medical Research and Education.

Rozsah diplomové práce: **doporuč. 50 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce: **Doc. Dr. Ing. Jana Klečková**
Katedra informatiky a výpočetní techniky

Datum zadání diplomové práce: **8. září 2023**
Termín odevzdání diplomové práce: **16. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2023

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického 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 Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 14. května 2024

. . . . . .
Josef Bozděch

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

V diplomové práci jsou diskutovány analytické a statistické možnosti rozšíření datového úložiště Data Lakehouse. Nejdříve je zkoumán koncept úložiště data lakehouse, následně jsou popsány již existující implementace tohoto úložiště - Databricks a koncept Data Lakehouse, na který je v této práci navazováno. Následně jsou zkoumána dostupná data pro Data Lakehouse z platformy MRE ZČU zabývající se shromažďováním medicínských dat. Následně jsou navrhnuty možnosti rozšíření Data Lakehouse, práce pokračuje popisem jejich implementace, výběr technologií a problémy, na které bylo narazeno při implementaci těchto návrhů. Nakonec je implementace otestována, výsledek práce porovnán s MRE, navrženy další možnosti rozšíření Data Lakehouse, shrnuty a vyhodnoceny dosažené výsledky.

Abstract

The thesis discusses the analytical and statistical possibilities of extending the Data Lakehouse data warehouse. First, the concept of Data Lakehouse is examined, then existing implementations of this repository - Databricks and the Data Lakehouse concept, which is built upon in this thesis, are described. Subsequently, the available data for Data Lakehouse from the MRE platform of ZČU dealing with medical data collection is examined. Subsequently, options for extending the Data Lakehouse are proposed, followed by a description of their implementation, technology selection and the problems encountered in implementing these proposals. Finally, the implementation is tested, the result of the work is compared with the MRE, further options for extending the Data Lakehouse are proposed, and the results are summarized and evaluated.

Klíčová slova

Delta Lake • Apache Spark • Vizualizace a statistika • Datová analýza • Preact • Zpracování informací • Data Lakehouse

Poděkování

Chtěl bych mojí vedoucí Doc. Dr. Ing. Janě Klečkové poděkovat za mnohonásobné rady a pomoc během tvorby této diplomové práce i během celého studia.

Obsah

Úvod 1

Množství dat ve všech oborech lidské činnosti exponenciálně roste. Big Data dopadají na velkou část z nás. Tato data se často pouze ukládají, práce s nimi je nicméně častokrát velice náročná a bez jejich předchozího pochopení skoro nemožná. Tato data často nemají předdefinovanou strukturu a k jejich ukládání se využívají tzv. datová jezera. Diplomová práce navazuje na projekt Data Lakehouse, který se zabývá implementací funkcionalit poskytovaných frameworkem Delta Lake. Práce se zaměřuje na porozumění novému archetypu databází zvaných data lakehouse a jejich použití. Bude popsáno, z jakých částí se data lakehouse skládá, formáty pro ukládání dat a jak je možné tato data do lakehouse nahrávat a jak je extrahovat. V další kapitole se práce zaměří na popis aplikace Data Lakehouse, která vznikla v posledních letech a snaží se architekturu lakehouse implementovat právě s pomocí frameworku Delta Lake. Následně jsou popsány datové kolekce, se kterými bylo v předchozí práci pracováno a se kterými se také pracuje v aplikaci Medical Research and Education. Poté bude popsán význam datové analýzy a statistik při výzkumu medicínských dat. Budou také navrženy metody analýzy vhodné zejména pro medicínská data.

V kapitole 6 budou diskutovány nedostatky aplikace a jak tyto nedostatky řešit v následující části. V sedmé kapitole pak budou navržená vylepšení implementována a popsán postup při implementaci. Aplikace bude nakonec uživatelsky a funkčně otestována. Bude také porovnána s dosavadními výsledky aplikace Medical Research and Education.

Koncept Data Lakehouse

2

Databáze prošly za dobu svojí existence dlouhým vývojem. Z doby, kdy se za databáze považovaly kartotéky, se přešlo v 60. letech 20. století k děrným štítkům a brzy následovaly první elektronické databáze. Následoval vznik prvních relačních databází, které se na dlouhou dobu staly dominantním typem databází. Tento trend se mění.

Data se dělí na strukturovaná a nestrukturovaná. Právě množství nestrukturovaných dat exponenciálně roste a relační databáze nejsou vhodným nástrojem pro jejich uchovávání. Tento trend umožnil rozšíření alternativních databází, které jsou pro tento typ dat určeny. Mezi dnes rozšířené technologie můžeme považovat NoSQL, grafové, nebo dokumentové databáze.

Pro ukládání těchto velkých dat se ale jako vhodný nástroj našel koncept zvaný datové jezero. Datová jezera jsou typ databází určených pro rychlé ukládání strukturovaných i nestrukturovaných dat.

Data se nyní mohou rychle a efektivně ukládat, problémem je ale jejich zpracování, bez nějž jsou data samotná k ničemu - nedají se použít kromě jejich zpětného načítání k datové analýze.

Tento problém řeší datová skladiště. Jejich cílem je extrakce významných dat, jejich sumarizace, agregace a interpretace. Uživatel pravděpodobně nedokáže z velkého množství záznamů interpretovat jejich význam, jejich agregaci naopak již pravděpodobně rozumět bude.

Z těchto dvou technologií - datového jezera a datového skladiště vznikl koncept zvaný Data lakehouse. V této kapitole je popsáno, jak data lakehouse funguje, jaká je jeho architektura a jak se dále pracuje s uloženými daty.

Lakehouse je nová, otevřená architektura, která kombinuje nejlepší prvky datových jezer a datových skladů. Lakehousy jsou umožněny novou koncepcí systému: implementací podobných datových struktur a funkcí správy dat jako v datovém skladu přímo nad levným cloudovým úložištěm v otevřených formátech. Sloučení datových jezer a datových skladů do jednoho systému znamená, že datové týmy mohou pracovat rychleji, protože mohou používat data bez nutnosti přístupu k více

systémům. Úroveň podpory jazyka SQL a integrace s nástroji BI u datových jezer je obecně dostatečná pro většinu podnikových datových skladů. K dispozici jsou materializované pohledy a uložené procedury, ale uživatelé mohou potřebovat použít jiné mechanismy, které nejsou ekvivalentní těm, jež se nacházejí v tradičních datových skladech [1].

2.1 Datová jezera

Datová jezera vznikla jako reakce na nedostatky datových skladů. Datové sklady sice poskytují vysoce výkonnou analytiku, ale nedokážou zvládnout všechny případy použití. Datové jezero představuje masivně škálovatelné úložiště, které uchovává velké množství nezpracovaných dat v jejich nativním formátu a zpracovává je, když je to potřeba. Tento typ úložiště je určen k práci s rozsáhlými a rychle příchozími nestrukturovanými a polostrukturovanými daty, jako jsou obrázky, video, audio a dokumenty, které jsou pro dnešní případy použití strojového učení a pokročilé analytiky klíčové. Analytické aplikace tak využívají data z datového jezera, která jsou okamžitě dostupná po jejich vytvoření.

Datová jezera se také odlišují od tradičních datových skladů svou schopností zachytávat data téměř v reálném čase, díky čemuž jsou vhodná pro rychle se měnící prostředí. Tato jezera mohou obsahovat sémantické databáze, které poskytují kontext a definují význam dat včetně jejich vztahů s dalšími daty. Datová jezera mohou kombinovat různé databázové přístupy, včetně SQL a NoSQL, a nabízí jak analytické, tak transakční zpracování.

Oproti hierarchickým datovým skladům, kde jsou data uložena ve složkách, mají datová jezera plochou architekturu, kde každý datový prvek má unikátní identifikátor a sadu rozšířených metadatových značek. Tento přístup nevyžaduje pevné schéma a umožňuje uchovávat data různého tvaru a velikosti s důrazem na zachování pořadí jejich příchodu. Datová jezera jsou koncipována jako centrální úložiště historických i nových dat, kde se schéma a požadavky na data definují až při dotazování [2].

2.1.1 Formáty pro ukládání dat v datových jezerech

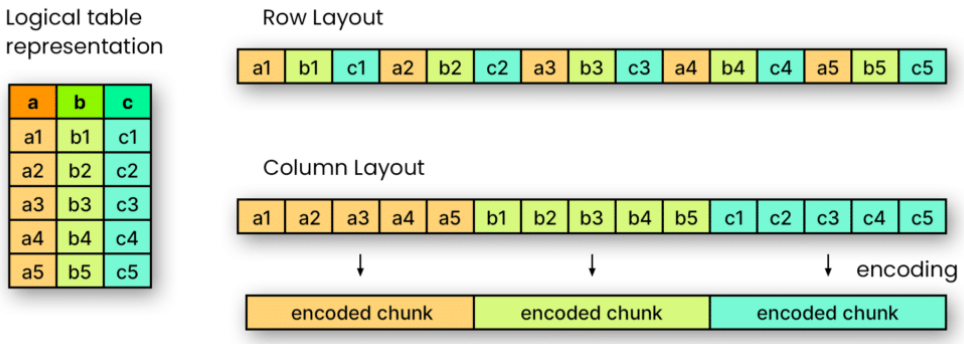
V následující části budou popsány nejčastější způsoby ukládání dat v datových jezerech. Každý z těchto formátů má vlastní výhody a nevýhody, které určují typ aplikací, pro které jsou jednotlivé formáty vhodné. Jedná se o:

- Parquet
- Avro
- CSV

2.1.1.1 Parquet

Formát souboru Parquet je binární formát používaný pro ukládání dat ve strukturovaném formátu. Je navržen tak, aby byl efektivní pro ukládání velkých objemů dat a umožňoval k nim rychlý přístup [3].

Apache Parquet je open source souborový formát, který ukládá data ve sloupcovém formátu (na rozdíl od řádkového formátu). Jako sloupcový formát pro ukládání dat nabízí několik výhod oproti řádkovým formátům pro analytické úlohy. Způsob ukládání dat je vidět na obrázku 2.1.



Obrázek 2.1: Ukázka formátu Parquet [3]

Výhody Apache Parquet: Řádkové formáty, jako jsou CSV a JSON, jsou čitelné pro lidi, zatímco sloupcové formáty jsou optimalizovány pro počítače. Jako sloupcový souborový formát mohou počítače číst Apache Parquet mnohem efektivněji a úsporněji než jiné formáty, což z něj činí ideální souborový formát pro ukládání velkých objemů dat, analytiku a datová jezera. Mezi hlavní výhody formátu Parquet patří vysoký výkon, účinná komprese a průmyslový standard.

Vysoký výkon: V Apache Parquet jsou hodnoty jednotlivých sloupců uloženy na disku společně. Protože analytické dotazy často potřebují pro operaci pouze podmnožinu sloupců, snižuje se tím množství dat, která je třeba načíst.

Soubory Parquet mají také statistiky o datech uložených v souboru, například minimální a maximální hodnoty dat sloupce v souboru a kolik řádků se v daném segmentu nachází. To umožňuje strojům vynechat celé segmenty nebo celé soubory podle toho, jakou část datového souboru úloha hledá.

Tyto dvě funkce vedou k výraznému zlepšení výkonu a snížení celkových nákladů díky snížení množství dat, která je třeba přečíst.

Účinná komprese: Apache Parquet podporuje vysoce účinnou kompresi. Mnoho kompresních kodeků je účinnějších, pokud komprimují podobná data. Sloupcový

formát technologie Parquet znamená, že sloupce podobných dat lze komprimovat společně, což vede k vyšší efektivitě.

Ukládání komprimovaných dat je nákladově efektivnější než ukládání nezpracovaných dat, takže použití nástroje Parquet může snížit náklady na ukládání velkých souborů dat. Komprimovaná data jsou také výkonnější než nekomprimovaná data, pokud je úzkým hrdlem vstupně-výstupní operace, což může být často případ analytických úloh, takže komprimovaná data v souborech Parquet mohou rovněž zvýšit výkon.

Průmyslový standard: Apache Parquet je standardní sloupcový formát souborů. Soubory Parquet lze číst téměř všemi dostupnými nástroji. To umožňuje používání více nástrojů.

Nevýhody aplikace Apache Parquet: Ačkoli je Apache Parquet standardním formátem souborů pro analytické úlohy, je třeba si uvědomit některé nevýhody. Různé pracovní zátěže a požadavky mohou v určitých situacích vést k použití jiného formátu.

Soubory v binární podobě nelze číst lidmi: Parquet je binární formát souborů optimalizovaný pro počítače, takže soubory Parquet nejsou přímo čitelné pro lidi. Soubor Parquet nemůžete otevřít v textovém editoru tak, jak je to možné u souboru CSV, a zjistit, co obsahuje. Existují nástroje, které převádějí binární reprezentaci na textovou, například `parquet-tools`, což ale znamená případný krok navíc při pokusu o čtení dat.

Pomalejší doba zápisu: Zápis souborů typu parquet může být pomalejší než u řádkových formátů, především proto, že obsahují metadata o obsahu souboru. Pro analytické účely jsou tyto pomalejší časy zápisu více než kompenzovány rychlými časy čtení. Nicméně v situacích, kdy je nejdůležitější čerstvost dat a latence událostí (např. v řádu desítek milisekund), se může vyplatit využít řádkový formát bez statistik, jako je Avro nebo CSV [3].

2.1.1.2 Parquet vs. CSV

CSV je jedním z nejrozšířenějších datových formátů. Zatímco soubory CSV se snadno otevírají pro lidskou kontrolu a někteří datoví analytici s velkými soubory CSV pohodlně pracují, použití Apache Parquet má oproti CSV mnoho výhod.

Parquet je vhodnější pro úlohy OLAP (analytické zpracování) než CSV. CSV je vhodný pro výměnu dat, protože je založen na textu a jedná se o velmi jednoduchý standard, ale sloupcová struktura a statistiky formátu Parquet umožňují dotazům zaměřit se na nejrelevantnější data pro analytické dotazy výběrem podmnožiny sloupců a řádků, které se mají číst. Naproti tomu řádkový formát, jako je CSV, vyžaduje čtení celého souboru, a pokud je tabulka složena ze souborů CSV, tak celé tabulky/oddílu.

Protože se dotazy provádějí pouze na podmnožinu sloupců, nikoli proti celému souboru dat, lze celé soubory vynechat, pokud to dotaz nevyžaduje. Formát Parquet má mnohem lepší kompresi a odpovědi jsou s ním mnohem rychlejší než s formátem CSV [3].

2.1.1.3 Apache Parquet vs. Apache Avro

Apache Avro je také binární souborový formát jako Parquet. Avro je však řádkový souborový formát, podobně jako CSV, a byl navržen pro minimalizaci latence zápisu. Soubory Avro mají mnohem méně řádků na soubor než Parquet, někdy dokonce jen jeden řádek na soubor. Nejběžnějším případem použití formátu Avro je streamování dat.

Parquet má oproti Avro stejné výhody jako CSV. Je lepší pro analytiku, má lepší výkon a je cenově výhodnější.

Výhody Avro oproti CSV Soubory Apache Avro mají schéma záznamů vložené v souboru. To je užitečné zejména při proudové analýze, například když se schéma aplikace produkující data může v průběhu času měnit.

Avro je také binární protokol, který poskytuje lepší výkon při čtení než CSV [3].

2.2 Datová skladiště

Datový sklad je typ systému pro správu dat, který je navržen tak, aby umožňoval a podporoval činnosti business intelligence (BI), zejména analytiku. Datové sklady jsou určeny výhradně k provádění dotazů a analýz a často obsahují velké množství historických dat. Data v datovém skladu jsou obvykle získávána z celé řady zdrojů, jako jsou soubory protokolů aplikací a transakční aplikace.

Datový sklad centralizuje a konsoliduje velké množství dat z různých zdrojů. Jeho analytické schopnosti umožňují organizacím získávat z dat cenné obchodní poznatky a zlepšovat tak rozhodování. Postupem času vytváří historické záznamy, které mohou být neocenitelné pro datové vědce a obchodní analytiky. Díky těmto schopnostem lze datový sklad považovat za "jediný zdroj pravdy"organizace [4].

2.2.1 Problémy s datovými skladišti

Datová skladiště jsou klíčové pro řadu podnikových procesů, avšak uživatelé často čelí problémům spojeným s nesprávnými daty, jejich zastaralostí a vysokými náklady. Jako řešení těchto výzev je prezentován koncept Lakehouse, nová generace otevřených platforem, která integruje datová jezera a sklady a umožňuje pokročilou analýzu. Tato platforma si klade za cíl odstranit náhodnou složitost existujících podnikových datových platforem a zefektivnit správu dat.

Prvním hlavním problémem, se kterým se setkávají uživatelé, je kvalita a spolehlivost dat, což je komplikováno současnými dvouúrovňovými datovými architekturami, které rozdělují data mezi jezera a sklady. Tyto systémy často vedou k rozdílům v sémantice podporovaných datových typů, dialektech SQL jazyka, rozdílných schématech a zvýšenému počtu úloh ETL/ELT, což zvyšuje pravděpodobnost výskytu chyb.

Druhým problémem je zpoždění dat způsobené oddělením oblasti pro ukládání příchozích dat od finálního úložiště, což vyžaduje použití periodických úloh ETL/ELT pro načítání dat. Ačkoli by teoreticky bylo možné implementovat více streamovacích potrubí pro rychlejší aktualizace, ty jsou často náročnější na správu než dávkové úlohy.

Třetím problémem je obtížná správa velkého množství nestrukturovaných dat, které se vyskytují v mnoha odvětvích a zahrnují obrázky, data ze senzorů, dokumenty atd. Tradiční SQL datové sklady a jejich API tyto typy dat často nepodporují efektivně.

Nakonec, stávající datové sklady a jezera nedokážou efektivně obsloužit aplikace strojového učení a datové vědy, protože tyto aplikace vyžadují zpracování velkých objemů dat pomocí kódu mimo SQL jazyk, což je přes standardní ODBC/JDBC rozhraní obtížné.

V důsledku těchto výzev se ukazuje, že poskytnutí přímého přístupu k datům v otevřeném formátu pro pokročilé analytické systémy se jeví jako nejefektivnější řešení. Aplikace strojového učení a datové vědy také potřebují řešení, které by přineslo výhody databázových systémů pro správu jejich dat, aby zajistily vyšší kvalitu, konzistenci a izolaci dat [5].

2.2.2 Databricks

Databricks, společnost zodpovědná za vznik Delta Lake provozuje vlastní produkt se stejným název Databricks. Tato platforma využívá kombinaci technologií Delta Lake, Apache Spark a mlflow k implementaci data lakehouse. Jedná se o cloudovou službu a je možné si vybrat, u kterého ze 3 poskytovatelů cloudových služeb se bude služba provozovat: Google Cloud, Microsoft Azure, nebo Amazon Web Services.

Delta Lake sám není lakehouse tak, jak je definovaný, je pouze nástrojem, jak jeho tvorby dosáhnout.

Pro práci s platformou Databricks je nutné mít založený účet. Je možné si vybrat, zdali se bude jednat o profesionální použití, možností je vybrat i tzv. komunitní edici, ta má omezené možnosti v její škálovatelnosti, pro osobní a výukové použití je však plně zdarma.

Na začátku je potřeba vytvořit server, na kterém bude nainstalována instance pro použití Databricks. Následně je možné začít nahrávat libovolná data ve formátech

CSV, JSON, nebo Avro. Tato data se ukládají do tabulek, tabulky je možné vytvářet dynamicky z přidaných dat. Data se následně dají zobrazovat, zobrazovat se může i jejich struktura.

2.2.2.1 Datová analýza v Databricks

Pro datovou analýzu v Databricks slouží nástroj `Notebook`. V tomto nástroji je možné programovat dotazy v jazyce Python, pro komunikaci s databází a jejími tabulkami slouží framework PySpark od společnosti Apache.

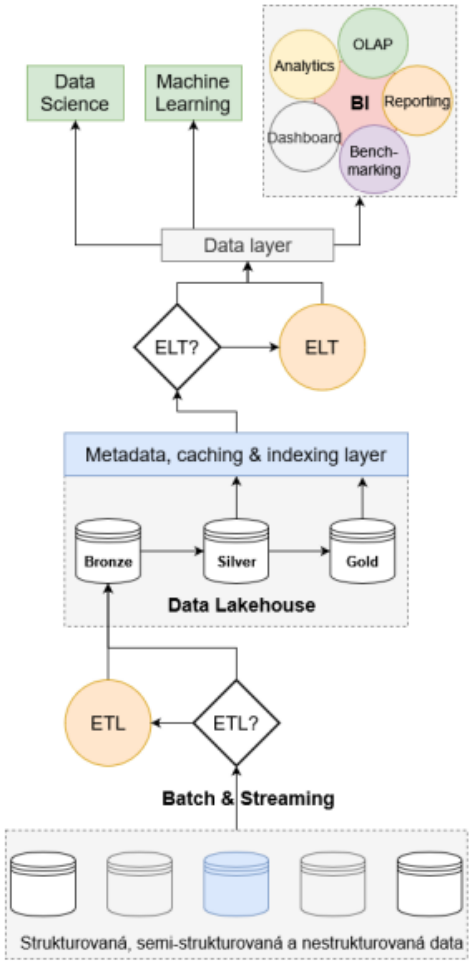
2.3 Architektura Lakehouse

Diagram 2.2 zobrazuje prakticky využívanou architekturu Data Lakehouse, do které vstupují strukturovaná, semi-strukturovaná a nestrukturovaná data. Tyto data často přicházejí ve velkých objemech buď v dávkách nebo kontinuálně jako data stream. V této architektuře může být použit ETL proces k extrakci dat, ale častěji se data ukládají přímo ve své surové formě do první vrstvy, která je v Databricks označována jako **bronzová vrstva** a u AWS jako přijímací vrstva. Během nahrávání se k datům přidávají relevantní metadata, což umožňuje transakční zpracování. K datům se přistupuje prostřednictvím API, které interaguje přímo s transakční vrstvou metadat. Často se zde uplatňuje metoda zvaná schema-on-read, která umožňuje použití procesu ELT podle potřeb nástrojů přistupujících k datům. Tyto nástroje pak mohou pracovat s daty v jejich nativním formátu. Tento model je typicky využíván týmy v oblastech strojového učení nebo vědecké analýzy dat. Data Lakehouse rovněž komunikuje s informačními systémy BI, přičemž pomocí dotazovacích jazyků jsou generovány analytické přehledy a reporty, které jsou prezentovány prostřednictvím dashboardů [6].

Delta Lake automaticky odmítne záznamy, které porušují tato očekávání, nebo je umístí do speciálního souboru mimo tabulku v datovém úložišti. Tyto jednoduché funkce jsou velmi užitečné pro zlepšení kvality dat založených na datovém jezeře. Vrstvy metadat jsou přirozeným místem pro implementaci funkcí správy, jako je řízení přístupu a protokolování auditů. Vrstva metadat může například zkontrolovat, zda má klient povolen přístup k tabulce, než mu udělí pověření ke čtení surových dat v tabulce z cloudového úložiště objektů, a může spolehlivě zaznamenávat všechny přístupy [5].

2.4 ETL a ELT

"Extract, Transform, Load"a "Extract, Load, Transform"jsou 2 protichůdné přístupy při nahrávání a extrakci dat v databázích. Jedná se o rozdílné přístupy implementace



Obrázek 2.2: Data Lakehouse architektura [6]

tzv. Datové pumpy. Datové pumpy získávají data ze vzájemně nekompatibilních zdrojů a také je transformují do nových odpovídajících struktur a následně ukládají do datového skladu nebo datového jezera. Data jsou tak připravena k pozdějším analýzám. Celý proces zabezpečuje také "čištění"dat, což představuje nejdůležitější úkol. Kvalita datových pump tedy přímo ovlivňuje kvalitu dat uložených v datovém skladu, a tím pádem také kvalitu informací, které z něj můžeme získat [7].

Oba procesy se skládají ze 3 kroků:

- Extrakce - Extrakce znamená vytáhnutí zdrojových dat z původní databáze nebo zdroje dat. Při ETL se data přesunou do dočasné odkládací oblasti. V případě ELT se data okamžitě přesunou do datového jezera nebo systému pro ukládání datového skladu.

- Transformace - Transformací se rozumí proces změny struktury a formátu

informací tak, aby se integrovaly s cílovým datovým systémem a ostatními daty v tomto systému.

- Načítání - Načítáním se rozumí proces ukládání informací do systému pro ukládání dat [8].

Každý z přístupů má vlastní využití a pro správný výběr je potřeba pochopit jejich výhody a nevýhody. V následující části budou tyto přístupy popsány a porovnány.

2.4.1 ETL

Datové sklady pro online analytické zpracování (OLAP) - ať už cloudové nebo onsite - musí pracovat s relačními datovými strukturami založenými na SQL. Proto musí být veškerá data načítaná do datového skladu OLAP, transformována do relačního formátu, než je datový sklad může přijmout. Součástí tohoto procesu transformace dat může být také mapování dat, které slouží ke spojení více zdrojů dat na základě vzájemně korelujících informací. To proto, aby platforma business intelligence mohla analyzovat informace jako jeden integrovaný celek.

Proto některé typy datových skladů vyžadují ETL - protože transformace musí proběhnout před načtením dat.

ETL vyžaduje nepřetržitý, průběžný proces s přesně definovaným pracovním postupem. ETL nejprve extrahuje data z homogenních nebo heterogenních zdrojů dat. Poté data uloží do odkládací oblasti. Odtud data procházejí procesem čištění, jsou obohacována a transformována a nakonec jsou uložena v datovém skladu [8].

2.4.2 ELT

ELT je zkratka pro "Extract, Load, and Transform". V tomto procesu dochází k transformaci dat po jejich načtení do řešení pro ukládání dat. ELT mohou využívat cloudová řešení datových skladů, pro všechny různé typy dat - včetně strukturovaných, nestrukturovaných, polostrukturovaných a dokonce i surových typů dat.

Proces ELT také funguje s datovými jezery. Před analýzou dat pomocí platformy business intelligence je ještě nutná transformace dat. K čištění, obohacování a transformaci dat však dochází až po jejich načtení do datového jezera. Zde je několik podrobností, kterým je třeba porozumět v souvislosti s ELT a datovými jezery:

ELT je relativně nová technologie, kterou umožnily moderní technologie cloudových serverů. Cloudová datová úložiště nabízejí téměř neomezené možnosti ukládání dat a škálovatelný výpočetní výkon. Například platformy, jako jsou Amazon Redshift a Google BigQuery, umožňují díky svým neuvěřitelným zpracovatelským schopnostem vytvářet ELT pipelines.

ELT ve spojení s datovým jezerem umožňuje okamžitě přijímat stále se rozšiřující zásobu nezpracovaných dat, jakmile jsou k dispozici. Není nutné data před uložením do datového jezera transformovat do speciálního formátu.

ELT transformuje pouze data potřebná pro konkrétní analýzu. Ačkoli to může zpomalit proces analýzy dat, nabízí to větší flexibilitu - protože můžete data transformovat různými způsoby za běhu a vytvářet tak různé typy metrik, prognóz a reportů. Naopak v případě ETL může být nutné upravit celý postup ETL - a strukturu dat ve skladu OLAP - pokud dříve rozhodnutá struktura neumožňuje nový typ analýzy.

ELT má specifičtější případy použití než ETL. Je důležité poznamenat, že nástroje a systémy ELT se stále vyvíjejí, takže nejsou tak spolehlivé jako ETL ve spojení s databází OLAP. Ačkoli je třeba vynaložit více úsilí na nastavení, ETL poskytuje přesnější poznatky při práci s obrovskými soubory dat [8].

2.5 Porovnání ETL vs. ELT

V rámci procesu ETL dochází ke zpomalení načítání dat v důsledku jejich transformace před samotným načtením. Skrze ELT je možné data načítat a transformovat simultánně.

Přístup ELT také umožňuje uchování nezpracovaných dat, což vede k vytvoření historického archivu, který lze využívat pro tvorbu analýz (business intelligence). To umožňuje týmům BI přizpůsobit své strategie a cíle tím, že umožňuje opakované vyhledávání a vývoj nových transformací v surových datech využívajících rozsáhlé datové sady. Na rozdíl od toho, ETL obvykle nevytváří kompletní sady surových dat, které by byly dostupné pro neomezené prohledávání.

Tyto charakteristiky činí ELT flexibilnější, efektivnější a lépe škálovatelným, zejména pro zpracování velkých objemů dat, které zahrnují jak strukturovaná, tak nestrukturovaná data, a pro vývoj různorodých obchodních informací.

V současnosti představuje ELT preferovanou metodu pro zpracování polostrukturovaných a nestrukturovaných dat, zatímco ETL se častěji používá pro strukturovaná data. Většina dat v moderním digitálním světě je nestrukturovaná (například obrázky, videa, PDF soubory, dokumenty PowerPoint), a přestože je zpracování takových dat složité, očekává se, že budoucnost přinese snahy o zlepšení jejich interpretace, kde ELT bude hrát klíčovou roli.

Naopak ETL je vhodnější pro situace, které vyžadují náročné výpočetní transformace, systémy se starší architekturou nebo pro pracovní postupy, kde je nutné manipulovat s daty před jejich vstupem do cílového systému, jako je například anonymizace osobních identifikačních údajů.

Obě metody, ETL i ELT, zahrnují procesy čištění a filtrování, což jsou zásadní kroky v procesu transformace dat. Vzhledem k tomu, že ETL dokončí transformaci

před načtením dat na server, je tato metoda vhodnější pro splnění norem shody a bezpečný přenos citlivých informací [8].

Popis aplikace Data Lakehouse

3

Data Lakehouse je aplikace postavená na technologii `Delta Lake`. Aplikace byla tvořena jako diplomová práce a měla za účel ověřit koncept Data Lakehouse a jeho práci s medicínskými daty ze systému MRE. V práci bylo úspěšně implementováno datové úložiště a rozhraní pracující s ním. V rámci seznámení se s dosavadní prací byl projekt spuštěn a prozkoumána struktura projektu. V následující sekci budou povrchně popsány jednotlivé části aplikace. Pro podrobný popis předchozí práce je potřeba přečíst diplomovou práci Koncept Data Lakehouse pro zpracování medicínských dat od Ing. Lukáše Moučky [6].

Aplikace splňuje vlastnosti třívrstvé architektury nazývané `MVC` (Model-View-Controller). Tato architektura dělí aplikaci na tři logické části, aby každá z nich mohla být upravována samostatně bez dopadu na ostatní [6].

3.1 Technologie

V následující sekci budou popsány technologie, které byly v projektu použity. Nebudou popsány jednotlivé knihovny, které aplikace využívá, `Apache Spark` je považován za jednu technologii, z níž jsou v projektu využity další knihovny.

3.1.1 Spring Boot

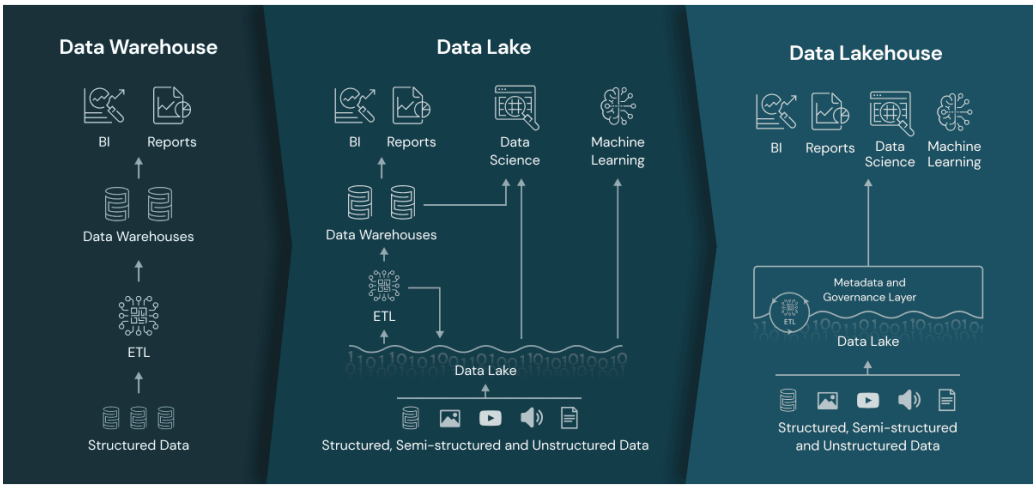
`Spring Boot` je framework založený na jazyce Java a umožňuje rychle vytvářet webové aplikace. `Spring Boot` je navržen tak, aby facilitoval proces vývoje samostatných a provozuschopných aplikací založených na Springu, které jsou připraveny k okamžitému spuštění. Tento framework přijímá určitý konceptuální postoj k využití platformy `Spring` a externích knihoven, čímž umožňuje uživatelům rychleji zahájit práci s minimem počátečních obtíží. Aplikace vytvořené pomocí `Spring Boot` vyžadují pouze základní konfiguraci frameworku `Spring`, což podstatně zjednodušuje proces vývoje.

3.1.2 Delta Lake

`Delta Lake` je open-source úložný framework, který umožňuje vytvářet formátově agnostickou architekturu Lakehouse s výpočetními enginy včetně `Spark, Google BigQuery, Databricks a API pro Scala, Java, Rust a Python` [9].

Na tomto je frameworku je položen základ pro tvorbu Data Lakehouse, neboť spravuje ukládání dat. Dalšími příklady frameworků pro tvorbu `Data Lakehouse` jsou například `Apache Hudi`, nebo `Apache Iceberg`.

Pro ukládání dat v datovém jezeře je možné využívat několika formátů, které byly popsány v kapitole 2.1.1. Delta Lake však podporuje pouze formát **Parquet**, který je ale pro účely tohoto úložiště vhodný, neboť aplikace `Data Lakehouse` má podporovat datovou analýzu, která zpravidla používá pouze některé vybrané sloupce z tabulky. Právě z tohoto důvodu nevadí oproti ostatním formátům pomalejší zápis do tabulek, naopak rychlost čtení dat je pro tento typ aplikace výhodou. Architektura lakehouse je vizualizována na obrázku 3.1.



Obrázek 3.1: Data Lakehouse architektura podle Databricks [10]

3.1.2.1 Datové vrstvy podle Databricks

Architektura lakehouse má podle společnosti Databricks 3 vrstvy. Každá tato vrstva plní v datovém úložišti jinou úlohu.

1. **Bronzová vrstva:**
    - Zachytává nezpracovaná provozní data z externích zdrojů pomocí OLTP systémů.

- Data jsou přijímána přes pipeline (např. Apache Kafka, Amazon Kinesis) nebo přímo z úložišť (např. Amazon S3).
- Při ukládání prochází data mírnou transformací a jsou konvertována do formátu Delta Lake, zachovávajíce původní strukturu tabulek.
- Umožňuje rychlé zachycení změn dat a poskytuje historický archiv zdrojů.

2. **Stříbrná vrstva:**
   - Obsahuje čistá atomická data zpracovávaná prostřednictvím OLAP systémů.
   - Data prochází rozsáhlejší transformací s důrazem na vynucení schémat a jsou strukturována do funkčních oblastí.
   - Provádění kompresních technik zlepšuje výkon dotazů.
   - Data jsou využívána pro pokročilé analýzy a strojové učení, přičemž proces ELT umožňuje minimální úpravy až do momentu potřeby.
3. **Zlatá vrstva:**
   - Data ze stříbrné vrstvy jsou transformována pomocí komplexních transformačních a obchodních pravidel.
   - Organizace dat probíhá ve specifických databázích.
   - Vrstva je určena především pro prezentaci dat, kde jsou data více denormalizovaná a používají se hvězdná schémata [11].

3.1.2.2 Limitace v oblasti relací v Delta Lake

Jedním z důležitých limitujících faktorů datových jezer je absence relací mezi jednotlivými tabulkami. Důvodem pro absenci relací je to, že se nejdená o klasickou relační databázi, jakými jsou například `MySQL` nebo `PostgreSQL`, nýbrž o skladiště dat, na které jsou kladeny jiné nároky. Zatímco ve standardních databázích jde zejména o integritu dat, úlohou datových jezer je uložit co nejefektivněji co nejvíce dat v co nejkratším čase.

3.1.3 Apache Spark

`Apache Spark` je open source analytický engine používaný pro práci s velkými objemy dat. `Spark` poskytuje nativní vazby pro programovací jazyky `Java, Scala, Python a R`. Kromě toho obsahuje několik knihoven pro podporu vytváření aplikací pro strojové učení, zpracování datových toků a zpracování grafů. `Apache Spark`

se skládá z jádra `Spark Core` a sady knihoven. Využití `Apache Spark` má pro projekt mnoho výhod:

- Rychlost - `Spark` vykonává velmi rychle díky ukládání dat do mezipaměti při více paralelních operacích. Hlavní vlastností Sparku je jeho in-memory engine, který zvyšuje rychlost zpracování; při zpracování dat ve velkém měřítku je tak až 100krát rychlejší než MapReduce při zpracování v paměti a 10krát rychlejší na disku. Spark to umožňuje díky snížení počtu operací čtení/zápisu na disk.
- Podpora více paralelních požadavků - `Apache Spark` dokáže spouštět více paralelních požadavků, včetně interaktivních dotazů, analýzy v reálném čase, strojového učení a zpracování grafů. Jedna aplikace může bez problémů obsluhovat více požadavků.
- Zvýšená použitelnost - Díky schopnosti podporovat několik programovacích jazyků je dynamický. Umožňuje rychlé psaní aplikací v jazycích Java, Scala, Python a R; dává vám tak k dispozici celou řadu jazyků pro tvorbu aplikací.
- Pokročilá analytika - `Spark` podporuje dotazy SQL, strojové učení, proudové zpracování a zpracování grafů [12].

3.1.4 Thymeleaf

`Thymeleaf` je moderní šablonovací engine Java na straně serveru pro webové i samostatné prostředí.

Hlavním cílem `Thymeleafu` je přinést do vývojového procesu elegantní přirozené šablony - HTML, které lze správně zobrazit v prohlížečích a zároveň fungují jako statické prototypy, což umožňuje lepší spolupráci ve vývojových týmech.

Díky modulům pro `Spring Framework`, řadě integrací s vašimi oblíbenými nástroji a možnosti připojit vlastní funkce je `Thymeleaf` ideální pro moderní vývoj webových stránek v HTML5 JVM [13].

V aplikaci se Thymeleaf používá na vykreslování všech webových stránek. Díky tomu, že většina stránek slouží k zobrazování jednoduchých seznamů, nebo jako jednoduché formuláře k nahrávání tabulek, jejich záznamů, atd., není potřeba složitějšího dynamického zobrazování.

3.1.5 JavaScript

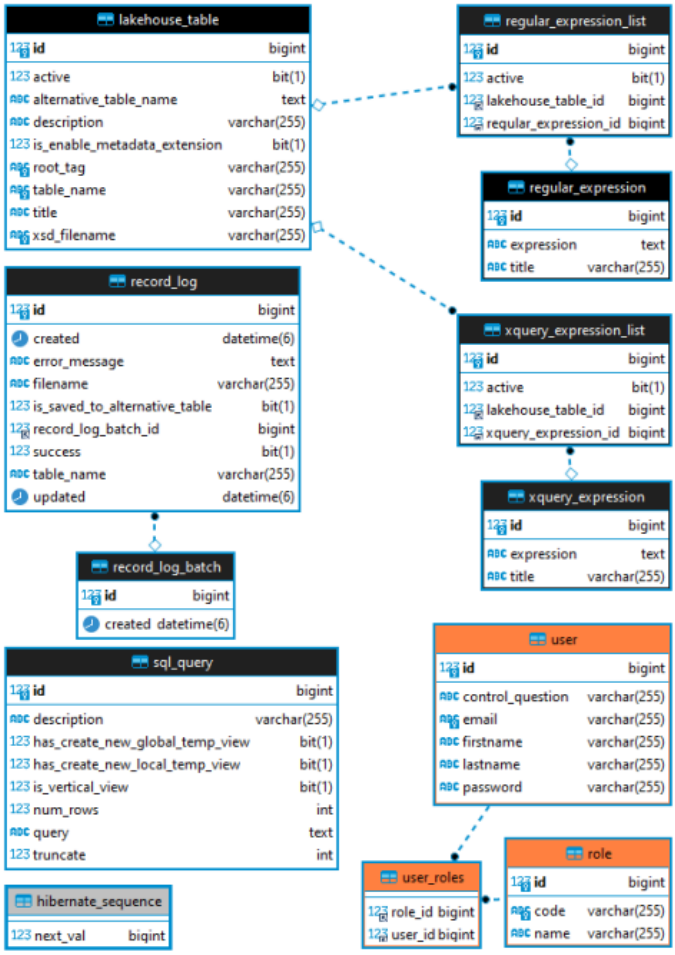
`JavaScript` je programovací jazyk, který se používá zejména při tvorbě webových stránek. V aplikaci se `JavaScript` využívá zejména v kombinaci s knihovnou jQuery.

`jQuery` je rychlá, malá a funkčně bohatá knihovna jazyka JavaScript. Díky snadno použitelnému rozhraní API, které funguje v mnoha prohlížečích, usnadňuje například procházení a manipulaci s dokumenty HTML, zpracování událostí, animace a AJAX. Díky kombinaci všestrannosti a rozšiřitelnosti změnila knihovna `jQuery` způsob, jakým miliony lidí píší JavaScript [14].

Největší roli hraje knihovna jQuery v odesílání REST požadavků na server.

3.1.6 MySQL

Aplikace využívá databázi MySQL k ukládání metadat o datech v `Delta Lake`. V databázi se také ukládají data o uživatelích, názvy všech tabulek, regulární výrazy a XQuery výrazy a logy o nahraných záznamech. V tabulce **sql_query** se také ukládají SQL dotazy, které si uživatel přál uchovat pro opakované použití. Schéma databáze se nachází na obrázku 3.2.



Obrázek 3.2: ERA diagram relační databáze [6]

3.2 Struktura kódu

Aplikace je sémanticky rozdělena na několik částí podle toho, jakou úlohu jednotlivé třídy v aplikaci plní.

- Controller - Třídy pro obsluhu vykreslování pohledů a pro obsluhu požadavků z View
- Service - Podpůrné služby pro plnění požadavků z vrstvy Controller
- Rest - Třídy pro obsluhu požadavků REST API.
- Support - Podpůrné třídy
- Dao - Data Access Object jsou třídy sloužící k obsluze získávání a ukládání dat
- Dto - Data Transfer Object. Jedná se o třídy určené pro přenos dat mezi serverem a klientem.
- Domain - Doménové třídy použité v Dao.
- WebApp - Složka, ve které je uložena webová část aplikace.
  - Thymeleaf - Jsou zde uloženy šablony pro vykreslování dat. Tyto šablony se načítají do tříd Controller. Pro každý Controller existuje vlastní šablona. Thymleaf je popsán v kapitole 3.1.4.
  - Scss - Kaskádové styly a importované kaskádové styly knihovny Bootstrap.
  - Js - JavaScriptové soubory obsluhující akce v klientovi. Může se jednat o akce odesílání dotazů serveru, nebo například dynamické vizualizace dat.

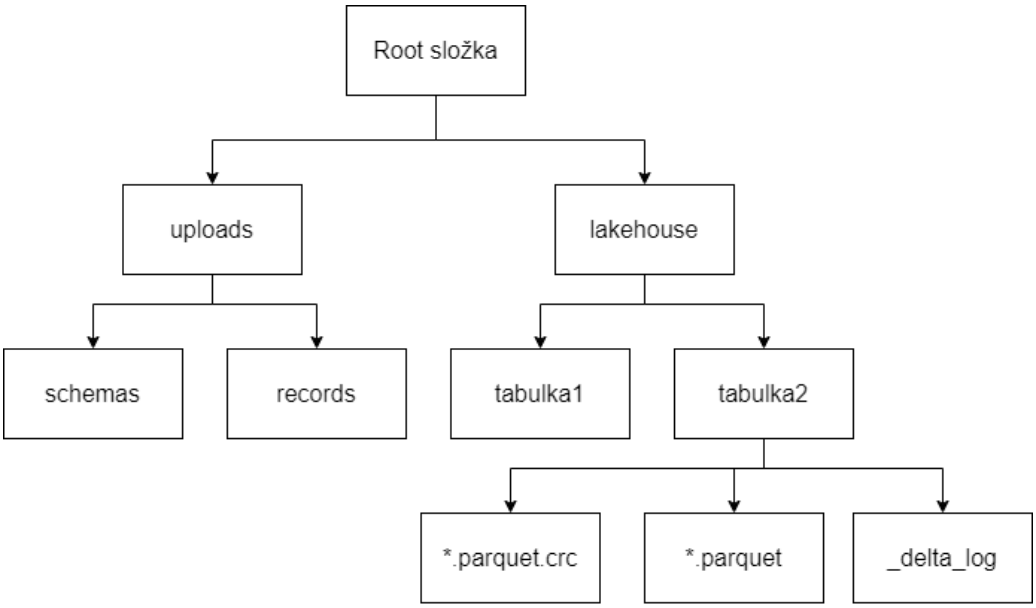
3.2.1 Načítání dat

Nejdůležitější funkcionalitou aplikace je načítání dat. Ty je možné načítat do jednotlivých tabulek, které musí být nejdříve vytvořeny. Aplikace umožňuje psát vlastní SQL příkazy a ovládat s nimi celý `Data Lakehouse`, cílem aplikace je ale oddělit uživatele od psaní vlastních SQL příkazů a umožnit mu pracovat na vyšší úrovni, bez velké znalosti jazyka SQL. Proto bude ve zbytku textu práce s SQL upozaděna a bude soustředěna na implementovanou funkcionalitu. Tabulky mohou být vytvořeny podle XSD šablony, ve kterých je uložena struktura dat zkoumaných v kapitole 4. Pro vytvoření tabulky je potřeba poskytnout základní informace, jako je název,

XSD soubor, možné je i nastavit alternativní tabulku, kam se aplikace pokusí uložit data, která neodpovídají šabloně dat, které se uživatel pokusí do tabulky vložit.

Po vytvoření tabulky je možné začít data nahrávat. Data je možné nahrávat ve formě XML souborů. Pro jednotlivé tabulky byla implementována možnost přiřazování `XQuery` a regulárních výrazů. Tyto výrazy umožňují upravovat data vstupu a mají za cíl zvyšovat propustnost dat v procesu nahrávání.

V kontextu Data Lakehouse je často zdůrazňován proces ELT, zatímco transformace dat při vstupu nejsou příliš zvažovány. Avšak praktické zkušenosti ukazují, že pro zlepšení průchodnosti záznamů systémem je nezbytné provádět alespoň základní transformaci dat na vstupu, což vyžaduje implementaci procesu ETL. Regulární výrazy jsou běžně používány k odstranění bílých znaků, prázdných a nesprávně spárovaných tagů a dalších podobných nedostatků v záznamech. Jedním z hlavních problémů bylo zacházení s tagy, které nemají žádnou hodnotu. Apache Spark tyto tagy neinterpretuje jako hodnoty null, ale jako prázdné řetězce, což může vést k tomu, že atributy ve výsledném `DataFrame` jsou klasifikovány jako datový typ `String`, což může způsobit nesrovnalosti s očekávaným schématem cílové tabulky `Delta` [6].



Obrázek 3.3: Struktura souborů v Data Lakehouse

Data se do aplikace nahrají, není však pro ně další využití. V aplikace není možné s daty kromě jejich extrakce nijak nakládat. Podle definice **Databricks** v kapitole 3.1.2.1 jsou tedy data uložena pouze na úrovni odpovídající definici **Bronzové vrstvy**. Struktura složek, do kterých se načtená data ukládají, je vidět na obr. 3.3. Do složky **schemas** se ukládají originální XSD soubory se schématy pro vytvoření

Delta tabulky. Složka **records** v sobě uchovává načtené datové soubory. Ve složce **lakehouse** se nachází podsložky s názvy jednotlivých tabulek, v nichž se nachází **parquet** soubory s daty jednotlivých tabulek a další metadata.

Dostupné datové kolekce v Medical Research and Education

4

Medical Research and Eduation je aplikace pro skladování dat. Aplikace se zaměřuje na skladování medicínských dat. Ukládají se zde například data týkající se mozkových mrtvic (RESQ), zánětlivých onemocnění střev (IBD) a další. Tato data jsou normalizována do formátu DASTA, který je popsán níže v této kapitole. K jednotlivým záznamům je také možné přidat obrazové záznamy ve formátu DICOM, který bude v této kapitole také popsán.

4.1 DASTA

DASTA je akronym pro Český národní datový standard pro výměnu informací ve zdravotnictví. Vydavatelem standardu je Ministerstvo zdravotnictví ČR. Koncepční základ standardu vytváří pracovní skupina datových standardů České společnosti zdravotnické informatiky a vědeckých informací (ČSZIVI) ČLS JEP ve spolupráci s ostatními subjekty podílejícími se na tvorbě standardu a dle potřeb Národního zdravotnického informačního systému.

Rozvoj standardu pro zdravotnické informační systémy byl iniciován v roce 1992 z důvodu neudržitelnosti individuálního propojování systémů, což bylo motivováno zkušenostmi několika vývojářů, kteří se setkali s existencí až třiceti různých protokolů a číselníků v jednom systému. Iniciativa Ministerstva zdravotnictví, lékařských fakult a soukromých firem vedla k vytvoření prvního datového standardu, který byl orientován na evropské standardy a normy.

První pracovní verze tohoto standardu, zvaná "Datový standard ministerstva zdravotnictví verze 1"(DS 1.0), byla publikována v roce 1994. Tato verze položila základy pro další rozvoj, ale neobsahovala komplexní datové bloky ani neřešila laboratorní komunikaci. Následný vývoj vedl k rozšíření datových bloků a k začlenění

laboratorního informačního systému, což vyústilo v vytvoření Národního číselníku laboratorních položek (NČLP) ve spolupráci s mezinárodní organizací IFCC.

Další verze, DS 2.01 a DS 3.01, přinesly rozšíření datových bloků a formalizovanou komunikaci s laboratorními systémy. Nejnovější verze, DS 4.01, zavedla koncepci klinických událostí, jejich formalizaci a identifikaci, a využila moderní XML technologie pro strukturování dat. Tyto verze postupně přecházely od textových formátů k využití možností XML schémat a jmenných prostorů[15].

Zdrojový kód 4.1: Zkrácená ukázka XSD souboru DASTA

```
<xs:schema attributeFormDefault="unqualified">
  <xs:element name="dasta">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="pm">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="a">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string"
                    name="jmeno"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute type="xs:string" name="id_soubor"/>
      <xs:attribute type="xs:string" name="verze_ds"/>
      <xs:attribute type="xs:dateTime" name="dat_vb"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Datový standard DASTA prochází konstantními úpravami, úprav zpravidla je několik ročně. Nejnovější verzí v době psaní této práce je verze DS 04.26.05, přičemž verze, se kterou je v práci operováno je verze DS 03.10.01 [15].

Ukázka XSD souboru s předpisem formátu DASTA se nachází v ukázce 4.1.

4.2 DICOM

DICOM (Digital Imaging and Communications in Medicine) je mezinárodní standard pro lékařské snímky a související informace. Definuje formáty pro lékařské snímky, které lze vyměňovat s daty a v kvalitě nezbytné pro klinické použití.

DICOM je implementován téměř ve všech radiologických, kardiologických zobrazovacích a radioterapeutických přístrojích (rentgen, CT, MRI, ultrazvuk atd.) a stále častěji i v přístrojích v dalších lékařských oblastech, jako je oftalmologie a stomatologie. Vzhledem k tomu, že se používají statisíce lékařských zobrazovacích zařízení, je DICOM jedním z nejrozšířenějších standardů pro zasílání zpráv ve zdravotnictví na světě. V současné době se pro klinickou péči používají doslova miliardy snímků DICOM.

Od svého prvního zveřejnění v roce 1993 způsobil DICOM revoluci v radiologické praxi a umožnil nahradit rentgenový film plně digitálním pracovním postupem. Od pohotovosti přes zátěžové testy srdce až po detekci rakoviny prsu je DICOM standardem, díky němuž lékařské zobrazování funguje pro lékaře i pro pacienty.

Standard DICOM je uznán Mezinárodní organizací pro normalizaci jako norma ISO 12052 [16].

4.3 Data použitá při vývoji aplikace

Data, která byla použita při implementaci a testování aplikace, jsou medicínská data ve standardizovaném formátu DASTA a týkají se konkrétně mozkových mrtvic, pocházejí z registru RESQ. Tato data se nacházejí ve formátu XML. Předpis jednotlivých tabulek je ve formátu XSD.

XSD je formát definice schématu pro XML dokumenty. Jedná se o standardizovaný způsob popisu struktury, obsahu a omezení platných hodnot v XML dokumentech. XSD je používán k definici datových modelů a validaci XML dokumentů na základě těchto modelů. XSD definuje následující klíčové prvky:

- **Elementy** - Elementy definují strukturu dat v XML dokumentu. Každý element může mít název a typ dat, který určuje povolené hodnoty a jejich formát.
- **Atributy** - Atributy jsou vlastnosti elementů, které mohou obsahovat další informace nebo metadata. Každý atribut má svůj název, typ a hodnotu.
- **Datové typy** - XSD obsahuje sadu základních datových typů, jako jsou řetězce, celá čísla, desetinná čísla, datumy a další. Tyto datové typy určují formát a validní hodnoty pro jednotlivé elementy a atributy.
- **Omezení** - XSD umožňuje definovat omezení pro elementy a atributy, jako jsou minimální a maximální délka řetězce, minimální a maximální hodnota čísla, regulární výrazy a další [17].

Pro vytvoření tabulek v Data Lakehouse existují soubory ve formátu XSD pro DASTA. Tyto soubory se nahrají pomocí webového rozhraní do aplikace, díky če-

muž se vytvoří odpovídající tabulky v Data Lakehouse. Následně je možné do těchto tabulek nahrávat záznamy, které jsou již uloženy ve standardních XML souborech, které odpovídají formátu předepsaném v XSD souborech.

Datová analýza a statistika

5

Statistika a datová analýza jsou klíčovými nástroji pro porozumění a interpretaci dat ve vědeckém výzkumu, průmyslu a mnoha dalších oblastech. Za prvé, statistika umožňuje extrahovat významné informace a vzory z dat, což umožňuje přijímat informovaná rozhodnutí. Analyzování dat může odhalit trendy, korelace a asociace mezi různými proměnnými, což může vést k objevování nových znalostí a nápadů. Díky statistickým metodám můžeme také provádět predikce a inferenční analýzu, což nám umožňuje odhadovat budoucí události a provádět závěry na základě vzorku dat.

Za druhé, statistika a datová analýza jsou důležité pro validaci hypotéz a výzkumných otázek. Pomocí statistických testů můžeme vyhodnotit, zda jsou pozorované rozdíly nebo vzory v datech statisticky významné nebo jsou pouhým náhodným jevem. To nám umožňuje potvrdit nebo vyvrátit naše výzkumné hypotézy a poskytuje důvěryhodné důkazy pro naše závěry. Statistika také poskytuje nástroje pro odhad chyb, intervalů spolehlivosti a kontrolu kvality dat, což je klíčové pro spolehlivost a důvěryhodnost vědeckých výzkumů a rozhodování v praxi. Statistika a datová analýza jsou nepostradatelnými nástroji pro získání hlubšího porozumění datům a přinášení informovaných rozhodnutí ve všech oblastech lidské činnosti.

5.1 Logistická regrese

Logistická regrese je statistická metoda používaná k modelování pravděpodobnosti binárních událostí. Jedná se o jednu z nejpoužívanějších metod klasifikace a predikce v oblastech, jako je medicína, ekonomie, sociologie a další.

Tento typ statistického modelu se často používá pro klasifikaci a prediktivní analýzu. Protože výsledek je pravděpodobnost, je závislá proměnná ohraničena mezi 0 a 1. V logistické regresi se na šance aplikuje logitová transformace - tj. pravděpodobnost úspěchu dělená pravděpodobností neúspěchu.

Logaritmické šance lze v rámci logistické regresní analýzy dat obtížně pochopit. V důsledku toho se běžně používají exponenciální odhady beta, aby se výsledky pře-

vedly na poměr šancí, což usnadňuje interpretaci výsledků. Poměr šancí představuje pravděpodobnost, že výsledek nastane při určité události, ve srovnání s pravděpodobností, že výsledek nastane, pokud by tato událost nenastala. Pokud je poměr šancí větší než 1, pak je událost spojena s vyšší pravděpodobností vzniku určitého výsledku. Naopak, pokud je poměr šancí (odds ratio) menší než 1, pak je událost spojena s nižší pravděpodobností výskytu daného výsledku [18].

5.1.1 Případy užití logistické regrese

- **Odhalování podvodů:** Logistické regresní modely mohou pomoci identifikovat anomálie v datech, které předpovídají podvody. Určité chování nebo charakteristiky mohou mít vyšší souvislost s podvodnými aktivitami, což je užitečné zejména pro bankovní a jiné finanční instituce při ochraně jejich klientů. Tyto postupy začaly používat také společnosti založené na SaaS, aby při analýze dat týkajících se výkonnosti podniku eliminovaly ze svých datových souborů falešné uživatelské účty.

- **Predikce odlivu klientů:** Specifické chování může svědčit o odlivu klientů v různých funkcích organizace. Například personalistické týmy a vedení mohou chtít vědět, zda ve společnosti existují vysoce výkonní pracovníci, u nichž hrozí riziko odchodu z organizace; tento typ poznatků může být podnětem k rozhovorům, jejichž cílem je porozumět problémovým oblastem ve společnosti, jako je kultura nebo odměňování. Případně se obchodní organizace může chtít dozvědět, kterým z jejich klientů hrozí odchod ke konkurenci. To může podnítit týmy k nastavení strategie udržení klientů, aby se zabránilo ztrátě příjmů.

- **Předpovídání nemocí:** V medicíně lze tento analytický přístup využít k předvídání pravděpodobnosti výskytu nemoci nebo onemocnění u dané populace. Zdravotnické organizace mohou nastavit preventivní péči pro osoby, které vykazují vyšší náchylnost k určitým nemocem [18].

Existuje mnoho dalších metod pro tvorbu analýzu a statistiku, dalším příkladem může být například K-Means, což je algoritmus pro tvorbu a shluků, což také může vést k predikci nemocí u pacientů. Právě předpovídání nemocí a korelace ukazatelů u pacienta je však důvod, proč byla logistická regrese vybrána jako analytická metoda pro implementaci v této aplikaci.

Návrh změn pro Data Lakehouse — 6

V předchozích kapitolách byly prozkoumány možnosti ukládání a zpracování dat v datových jezerech, skladištích a data lakehouse. Bylo zjištěno, že aplikace Data Lakehouse umožňuje tvorbu tabulek, vývoj jejich schémat a nahrávání dat do jednotlivých tabulek, významným nedostatkem této aplikace je však jakákoliv vizualizace získaných dat.

V kapitole 2 bylo také diskutováno, že lakehouse je kombinací datového jezera a datového skladiště. Datové jezero bylo v aplikaci úspěšně implementováno, data se správně ukládají do příslušných tabulek. Existuje dokonce i kontrola dat na vstupu ve formě regulárních výrazů a XQuery výrazů, což má za následek zvýšenou propustnost záznamů do datového úložiště.

Uživatel tedy data nahraje, kromě SQL příkazu je nemá však jak jinak vizualizovat a zkoumat. Funkce datového skladiště není naplněna. V průběhu analýzy byly stanoveny funkce, které jsou pro uživatele přínosem při datové analýze:

1. Umožnění nahrávání více formátů dat - Datová jezera plní úlohu úložiště pro libovolný typ dat. Aplikace Data lakehouse nicméně umožňuje ukládat pouze jediný typ dat - XML. Bylo by vhodné rozšířit možnosti o další typy.

2. Vizualizace datové struktury tabulky - znalost datové struktury je pro uživatele důležitá pro kontrolu formátu vstupních dat. Dalším případem, kdy je vizualizace datové struktury důležitá, je pro kontrolu vývoje schématu tabulky. Pokud uživatel umožňuje rozšiřování schématu tabulky, může se stát, že se do tabulky nahrají data v nesprávném formátu. Následně může zkontrolovat pomocí vizualizace schéma a kdy se změna stala.

3. Výpočet statistik - Statistiky jednotlivých tabulek jsou pro datovou analýzu významným nástrojem. Při pohledu na statistiky může uživatel rychle zkontrolovat, zdali počet nahraných záznamů odpovídá jeho očekávání, nebo zdali se v jednotlivých sloupcích nachází očekávané hodnoty, nebo jestli se do tabulky nahrávají prázdné záznamy. Díky statistikám je možné také např. odhalovat extrémní hodnoty.

4. Datová pumpa - Na vstupu, jak již bylo řečeno, existují základní filtry pro základní úpravu dat. Zde problém nastává v případě, že jsou na vstupu tabulky, které mají mezi sebou relace. Delta lake neumožňuje tvorbu relací ve smyslu klasických relačních SQL databází. Cílem je vytvořit mechanismus, který tuto funkci relačních databází bude replikovat. Dalším využitím datové pumpy je filtrace požadovaných řádek pro datovou analýzu, případně výběr požadovaných sloupců.

5. Analytické metody - Cílem datových skladišť je analýza dat a odvození důsledků ze zkoumání. Aplikace postrádá analytické metody, kterými by bylo možné data zkoumat.

Implementace funkcionalit

7

7.1 Dynamická vizualizace

Přechod od statického generování obsahu pomocí Thymeleaf k dynamickému renderování s využitím knihovny `Preact` přináší několik výhod, které mohou zlepšit efektivitu a uživatelskou zkušenost aplikace.

- Zlepšení práce s uživatelským prostředím - Tradiční metody jako Thymeleaf zpracovávají HTML na serveru a odesílají ho jako kompletní stránku klientovi. Tento přístup může vést k pomalé odezvě aplikace, jelikož uživatel musí čekat na načtení celé stránky při každé interakci. `Preact`, naopak, umožňuje dynamické renderování na straně klienta, což znamená, že aktualizuje pouze změněné části uživatelského rozhraní bez nutnosti znovu načítat celou stránku. Tím je dosaženo rychlejší a plynulejší interakce.
- Redukce zátěže serveru - Při použití `Preactu` je část zpracování přenesena ze serveru na klienta, což snižuje zátěž serveru. Tato redukce může výrazně zvýšit kapacitu serveru pro obsluhu více současných uživatelů a zlepšit škálovatelnost aplikace.
- Integrace s vizualizačními technologiemi - `Preact` je kompatibilní s širokou škálou současných `JavaScriptových` nástrojů a knihoven, což umožňuje vývojářům efektivněji implementovat nové funkce a využívat existující zdroje. Na rozdíl od `Thymeleafu`, který je primárně šablonovací nástroj na straně serveru, Preact umožňuje využití `JavaScriptových` knihoven, jako je `Chart.js` použitý v tomto projektu, k dynamické vizualizaci grafů a statistik.
- Podpora server-side renderingu (SSR) - Na rozdíl od základní knihovny `React`, `Preact` nativně podporuje server-side rendering (SSR), což je klíčové pro integraci do stávajících aplikací bez potřeby zavádění dodatečných kroků při sestavování aplikace. Pro nasazení klasického `Reactu` by bylo nutné přidat ba-

líčkovací nástroj jako `WebPack`. `Preact` tuto složitost eliminuje tím, že umožňuje renderování na serveru bez potřeby další infrastruktury.

7.1.1 Vizualizace statistik s pomocí grafů

Během práce bylo určeno několik bodů v aplikaci, kde je potřeba přidat grafovou vizualizaci dat, četnost nahraných záznamů, nebo například počet unikátních hodnot ve sloupci. Pro vykreslování grafů byly nalezeny 2 oblíbené `JavaScriptové` knihovny: **Chart.js** a **D3.js**.

7.1.1.1 Chart.js

`Chart.js` je knihovna poskytující 9 základních typů grafů, jako je sloupcový, bodový, liniový, nebo koláčový graf. Pro vstupní data pro každý typ grafu má přesně stanovené API, podle kterého se uživatel musí řídit. Práce s knihovnou je jednoduchá a má dostatečně podrobnou dokumentaci s příklady.

Ukázka 7.1 v sobě obsahuje kód pro vytvoření jednoduchého sloupcového grafu.

Zdrojový kód 7.1: Ukázka vytvoření naplněného sloupcového grafu [19]

```
1
2  const ctx = document.getElementById('myChart');
3
4  new Chart(ctx, {
5    type: 'bar',
6    data: {
7      labels: ['Red', 'Blue', 'Yellow', 'Green'],
8      datasets: [{
9        label: '# of Votes',
10       data: [12, 19, 3, 5, 2, 3],
11       borderWidth: 1
12     }]
13   },
14   options: {
15     scales: {
16       y: {
17         beginAtZero: true
18       }
19     }
20   }
21 });
```

7.1.1.2 D3.js

`D3.js` je druhou uvažovanou knihovnou v relaci s vykreslováním grafů.

D3.js je bezplatná open-source knihovna JavaScriptu pro vizualizaci dat. Její nízkoúrovňový přístup založený na webových standardech nabízí bezkonkurenční flexibilitu při tvorbě dynamické grafiky založené na datech [20].

Zdrojový kód 7.2: Ukázka tvorby prázdného 2D grafu

```
const width = 640;
...inicializace ostatních proměnných
const x = d3.scaleUtc()
    .domain([new Date("2023-01-01"),
    new Date("2024-01-01")])
    .range([marginLeft, width - marginRight]);
const y = d3.scaleLinear().domain([0, 100])
    .range([height - marginBottom, marginTop]);
const svg = d3.create("svg").attr("width", width)
    .attr("height", height);
svg.append("g")
    .attr("transform",
    `translate(0,${height - marginBottom})`)
    .call(d3.axisBottom(x));
svg.append("g").attr("transform",
`translate(${marginLeft},0)`)
    .call(d3.axisLeft(y));
return svg.node();
```

7.1.1.3 Porovnání knihoven

Během implementace funkcionalit aplikace byla porovnána práce s oběma knihovnami. Cílem bylo vytvořit stejný graf. Výsledkem porovnání je, že s knihovnou Chart.js se pracuje objektivně mnohem snáze, neboť umožňuje vývojáři pracovat na mnohem vyšší úrovni kódu, než knihovna D3.js. Ukázka 7.2 dokazuje složitost vytvoření jednoduchého prázdného grafu. V porovnání s Chart.js je vytvoření grafu mnohem náročnější. Proto je knihovna Chart.js použita ve zbytku aplikace.

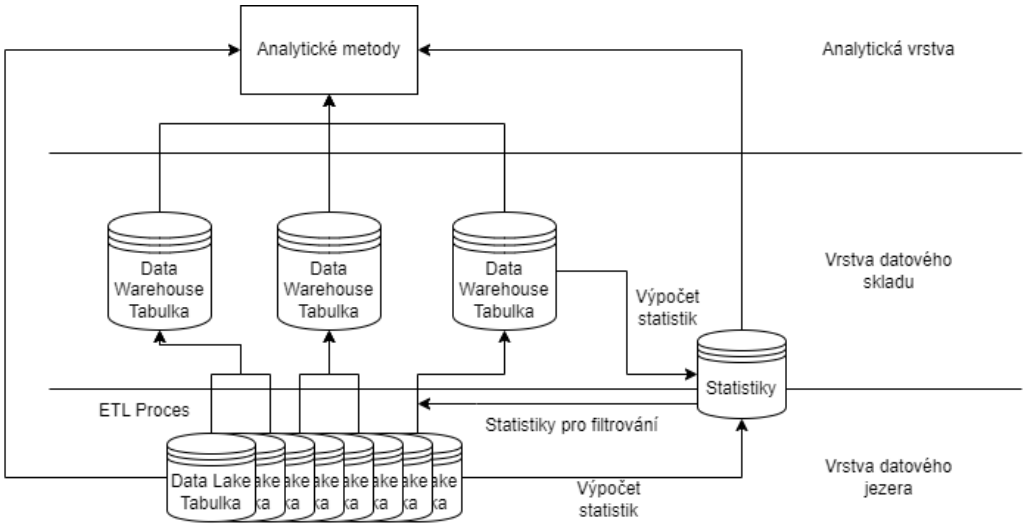
Knihovna D3.js má v aplikaci do budoucna nicméně velké opodstatnění. Jednou z možností, jak využít tuto knihovnu, je například vizualizace dat při vyhledávání v data dictionary a další vizualizační techniky při data miningu.

7.2 Tvorba nové datové vrstvy

Data, která přichází do tabulek nemusí být a zpravidla nejsou úplná a uzpůsobená pro datovou analýzu a výpočet statistik. V tabulce mohou být přebytečné sloupce, řádky s nullovými hodnotami, atd. Hlavním důvodem pro tvorbu této vrstvy je fakt, že tabulky, které mají mezi sebou libovolný typ relace (1:1, 1:N, M:N), nemají pro tyto relace žádnou podporu, jak již bylo zmíněno v kapitole 3.1.2.2. Statistiky vypo-

čítané pro samotné tabulky nemusí dávat smysl, bylo cílem relace alespoň částečně nahradit.

Vzhledem k tomu, že záznamy DASTA o pacientech mohou v různých časových rozpětích přicházet opakovaně a s jinými údaji, nejsou záznamy o uživatelích, kteří nemají ukončené pozorování, vypovídající. Tyto záznamy je potřeba pro účely analýzy vyfiltrovat. Obrázek 7.1 ukazuje rozdělení aplikace do jednotlivých vrstev, vrstva datového jezera představuje **bronzovou vrstvu** podle architektury Databricks, vrstva datového skladiště je ekvivalentem **stříbrné vrstvy**. **Zlatá vrstva** je zase reprezentována datovou analytikou, která je později také popisována. V následující části je popsán postup, jak je vrstva datového skladiště implementována.



Obrázek 7.1: Vizualizace procesu odvození tabulky

7.3 Odvozování tabulek

Z toho důvodu byla v aplikaci implementována datová pumpa, která tvoří na základě úkolu od uživatele nové, odvozené tabulky z podkladových tabulek. Tyto tabulky mohou být, stejně jako původní tabulky, použity k dalším analýzám.

K tomuto problému existují 2 řešení, každé z nich má jasné výhody a nevýhody:

- Tvorba odvozených tabulek - uživatel při požadavku na čistá data vytvoří novou tabulku, ve které budou data uložena. Výhody
  - Existující infrastruktura - pro tabulky již existují všechny potřebné funkce - ukládání, načítání, mazání

  - Dotazy se dají "zakonzervovat"- V případě, že uživatel chce dotaz provést k určitému momentu, lze zakázat další aktualizaci tabulky. Díky tomu může uživatel ke statistikám přistoupit k danému datu a zkoumat je mezi sebou.
  - Data zůstanou v tabulce i v případě, že budou podkladové tabulky smazány.

  - Větší velikost - pokaždé, když se vytvoří nová tabulka, jsou data kopírována. Kvůli tomu může dojít k extrémní expanzi úložiště.
  - Potřeba aktualizace dat - tabulky nejsou mezi sebou nijak propojeny, je potřeba sledovat po vytvoření odvozené tabulky podkladové tabulky a na základě změn v jejich datech aktualizovat i data v odvozené tabulce.

- Vytvoření pohledů - Pohledy jsou druhou možností pro zobrazování dat. Data z pohledů nejsou nikde ukládána a data v nich jsou stále aktuální.

  - Menší náročnost na úložiště - data není nikde potřeba ukládat, zmenší se tedy velikost úložiště
  - Data jsou vždy aktuální - Není potřeba implementovat žádnou pipeline, která by aktualizovala data v pohledu.

  - Tvorba nové části aplikace - s pohledy není nikde v aplikaci dosud pracováno. Časová náročnost této metody bude náročnější.
  - Ukládání a výpočet statistik - vzhledem k tomu, že nejsou data z pohledů nikde ukládána, je potřeba vzít v potaz při nahrávání záznamů do podkladových tabulek aktualizaci a ukládání statistik příslušných pohledů.
  - Při ztrátě podkladové tabulky jsou ztracena i vyčištěná data.

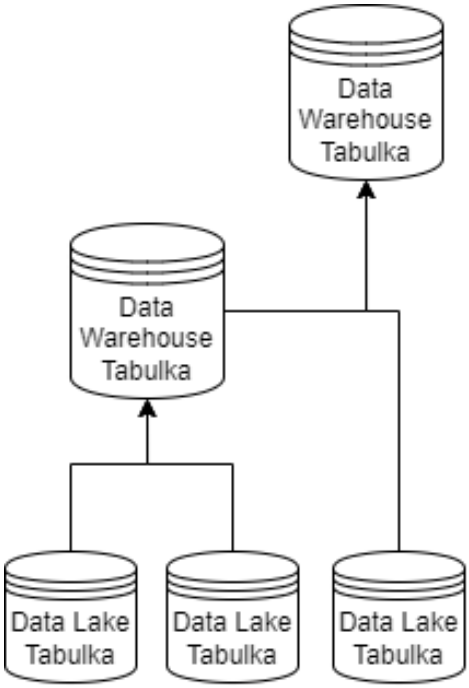
Z výše uvedených důvodů bylo rozhodnuto o implementace pomocí tabulek.

Pro uživatele byla vytvořena část s názvem "`Vytvoření odvozené tabulky`". Po kliknutí na tlačítko se mu zobrazí modalové okno, ve kterém je mu umožněno vytvořit novou tabulku. Uživatel musí vyplnit několik položek. Povinný je název a alespoň jedna tabulka. Další povinnou položkou jsou sloupce. Ve výchozím nastavení jsou všechny sloupce vybrány, kliknutím na checkbox Vybrat vše lze všechny sloupce hromadně zaškrtnout nebo odškrtnout. Pokud chce uživatel vytvořit novou tabulku pouze z jedné tabulky, může po tomto kroku kliknout na tlačítko "`Vytvořit`".

V případě, že chce uživatel vytvořit tabulku z více tabulek, je potřeba vybrat druhou tabulku a požadované sloupce.

Také je potřeba vybrat sloupec z každé tabulky, nad kterým bude provedena operace `JOIN`. SQL obecně umožňuje provádět JOIN nad více sloupci a je to běžně používáno, nicméně v rámci zjednodušeného provedení byla implementována operace JOIN s právě jedním sloupcem/klíčem. Posledním požadovaným parametrem je typ JOINu, který nabývá výchozí hodnoty INNER.

Architektura aplikace dovoluje skládat více tabulek dohromady. Pokud chce uživatel například vytvořit odvozenou tabulku ze 3 různých tabulek, musí nejdříve udělat JOIN nad 2 tabulkami, které vytvoří 1 odvozenou tabulku, poté udělá uživatel JOIN nad touto tabulkou a nad zbylou tabulkou. Tabulky, ze kterých jsou tabulky odvozeny, musí všechny v datovém jezeře zůstat. Pokud by se libovolná tabulka z řetězce odvozených tabulek smazala, již by nebylo možné aktualizovat data v odvozených tabulkách. Proces spojování více tabulek je vyobrazen na obr. 7.2.



Obrázek 7.2: Postup pro skládání více tabulek

7.3.1 Čištění dat

Jak již bylo zmíněno v předchozí části, pro analýzu velice významná funkcionalita čištění dat, nejen výběr požadovaných sloupců a spojování tabulek dohromady. Proto bylo umožněno pro každý sloupec vytvořit filtr, podle kterého budou data v nové tabulce filtrována. Filtrů existuje veliká řada, podmínek v SQL existuje příliš mnoho pro to, aby mohly být všechny pokryty v této práci. Proto byly vybrány

pouze nejvýznamnější podmínky, které jsou zde použity pro ukázku a je možno je později rozšířit. Pro každý sloupec je umožněno filtrování IS NOT NULL. To se dá nastavit kliknutím na checkbox ve sloupci vedle názvu filtrovaného sloupce. Jako nejdůležitější podmínky byly stanoveny podmínky datové typy STRING a NUMBER.

7.3.2 Filtrace řetězců

Filtrování řetězců je umožněno podle existujících řetězců ve sloupci. Tyto hodnoty je potřeba pro filtrování znát, hodnoty jsou proto získány ze statistik vybraných tabulek, jež byly předtím vypočítány. Uživatel si kliknutím na hodnoty vybere, které chce zahrnout, při filtrování je mezi nimi vytvořena podmínka OR.

7.3.3 Filtrace čísel

Pro čísla je využita podmínka BETWEEN. Stejně jako pro řetězce jsou využity vypočítané statistiky. Podmínek BETWEEN je možno opět zadat více a mezi nimi je relace OR.

Možné je také nechat jednu z možností nevyplněnou, pak se místo podmínky BETWEEN použije podmínka GREATER OR EQUAL THAN, případně LESSER OR EQUAL THAN.

7.3.4 Aktualizace odvozené tabulky

Protože jsou tabulky odvozovány z jiných tabulek, je potřeba zařídit čerstvost dat v odvozené tabulce. Data, která se načtou do podkladové tabulky, musí projít stejným ETL systémem, stejně jako při úvodní tvorbě tabulky. Možnosti pro nahrávání dat jsou 2:

- Přidávání nových dat na konec tabulky - nově příchozí data budou načtena na konec tabulky, tzv. "append"mód.
- Přepsání současných dat - data z tabulky budou kompletně smazána a následně budou zapsána data nová.

Při výběru metody načítání dat byla v úvahu brána cena a kvalita zapsaných dat. V případě, že jsou podmínky transformace v datové pumpě napsány ve formě, kdy podmínce může odpovídat více záznamů, postačuje pro transformaci připínání nových záznamů na konec tabulky. Ukázka 7.3 ukazuje SQL dotaz, ve kterém je podmínka, již může splňovat více záznamů v tabulce.

Zdrojový kód 7.3: Ukázka podmínky umožňující append mód

```
1 SELECT * FROM tabulka WHERE tabulka.sloupec = 'hodnota';
```

Při vývoji funkce nastal problém při zjištění, že může nastat podmínka, kdy chceme v tabulce některé záznamy pouze jednou. Při konzultaci případných filtrů bylo zjištěno, že případní uživatelé Data Lakehouse mohou s medicínskými daty **DASTA** zacházet tak, že chtějí filtrovat pouze nejnovější záznam pacienta. V případě, že by byla implementace provedena metodou append, nastane následující problém:

1. Uživatel vytvoří odvozenou tabulku.
2. Do podkladové tabulky přidá nový záznam pacienta.
3. Protože se jedná o nejnovější záznam daného pacienta, vloží se odvozené tabulky.
4. Uživatel zadá do podkladové tabulky další záznam stejného pacienta.
5. Opět se jedná o nejnovější záznam, filtr je splněn, záznam je vložen do odvozené tabulky.
6. V odvozené tabulce se nachází 2 záznamy jednoho pacienta.

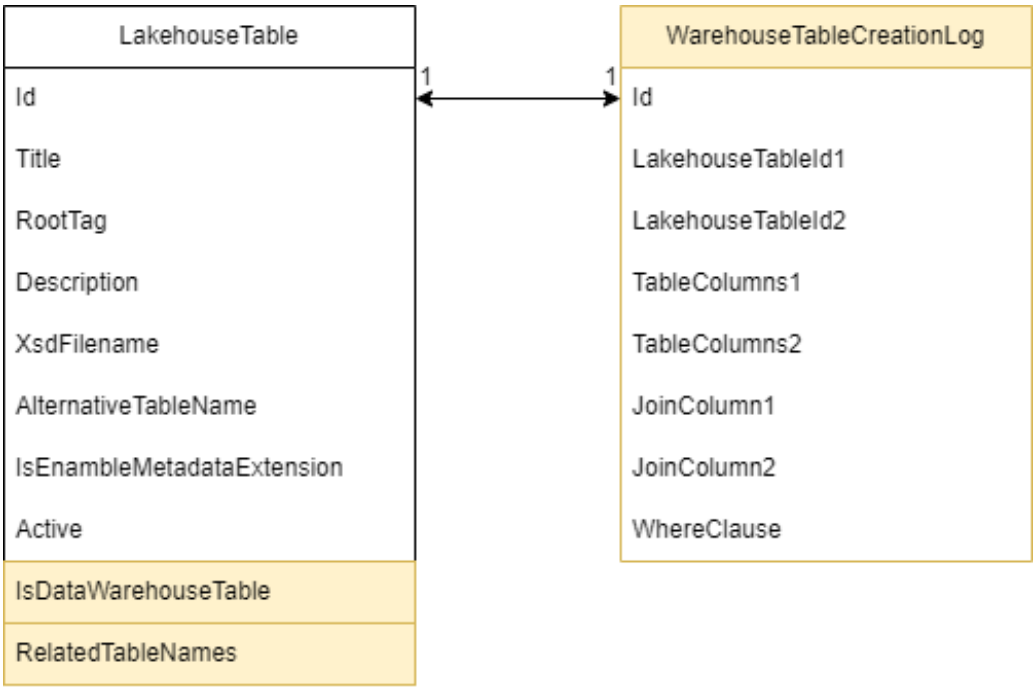
Kvůli tomuto příkladu bylo rozhodnuto o implementaci 2. možnosti. Data z tabulky se vždy smažou a projdou znovu celou datovou pumpou. Tím je zaručena integrita dat.

Jedná o velice drahou operaci, která může trvat v závislosti na velikosti dat jednotky až desítky minut, nicméně data v datovém skladišti nemusí být aktualizovaná okamžitě. Data se do datového jezera zpravidla nahrávají dávkově, datová analýza zpravidla není závislá na okamžitých datech, ale na dlouhodobých trendech.

Aplikace při vytváření odvozené tabulky musí uchovávat informace o tom, jak byla odvozená tabulka vytvořena. Metadata o postupu tvorby tabulky se ukládají do tabulky v MySQL. ERA model relace mezi informacemi o tabulce a metadaty o jejím vytvoření se nachází na obrázku 3.2. Tabulka `WarehouseTableCreationLog` slouží jako úložiště záznamů k rekreaci tabulky. Tabulka uchovává v jednotlivých sloupcích informace o tabulkách, ze kterých byla vytvořena, vybrané sloupce z každé tabulky, sloupce pro spojení a případně WHERE klauzuli pro SQL.

Každý záznam v tabulce je vázán na záznam v tabulce `LakehouseTable`. Relace mezi těmito 2 tabulkami je 1:1. Vzniklá vazba se nachází na obr. 7.3. Změny v modelu oproti původnímu jsou označeny žlutě.

Aby bylo možné při nahrávání nových záznamů do podkladových tabulek zjistit, jaké odvozené tabulky se na podkladovou tabulku vážou, byl do `LakehouseTable` přidán atribut `RelatedTableNames`, ve kterém se v řetězci odděleným čárkami uchovávají názvy tabulek, které byly z dané tabulky odvozeny.



Obrázek 7.3: Rozšíření ERA modelu

Poté, co se nahraje záznam, se zkontroluje, zdali existují vázané tabulky. Pokud ano, načte se příslušný `WarehouseTableCreationLog` záznam, podle kterého se odvozená tabulka přepíše.

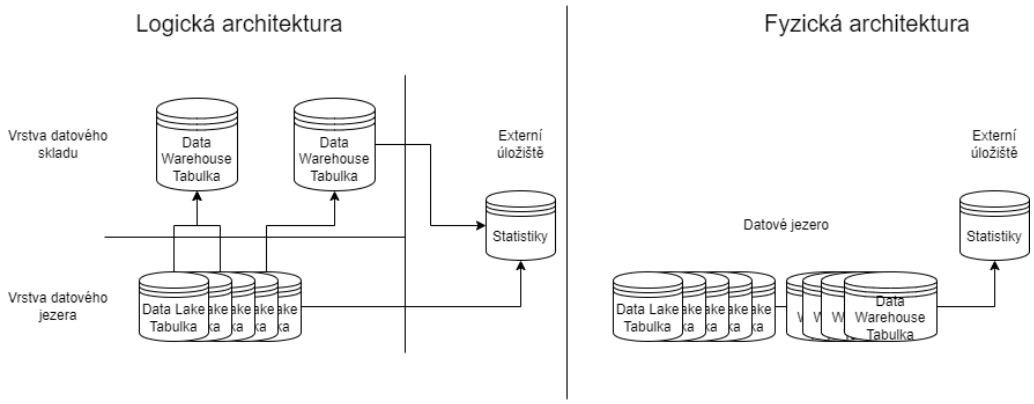
7.3.5 Architektura úložiště

Díky rozšíření popsanému v této sekci vznikl nový typ tabulek. Tyto tabulky jsou definovány z jiných tabulek, což je hierarchicky odděluje od původního typu tabulek.

Dosud byla logická a fyzická architektura úložiště stejná. Tabulky byly uloženy vedle sebe bez jakékoliv návaznosti na sebe. To se po rozšíření změnilo. Odvozené tabulky mají nyní relaci na ostatní tabulky. Tyto tabulky však zůstávají uloženy na stejném místě. Rozdíl mezi architekturami, který vznikl po této změně, je vidět na obrázku 7.4.

V logické architektuře existují 2 vrstvy: Vrstva datového jezera a vrstva datového skladu. V tomto rozdělení jsou odvozené tabulky na vyšší úrovni architektury, než tabulky podkladové. Tyto vrstvy odpovídají definici **bronzové** a **stříbrné** vrstvy od společnosti **Databricks**.

Ve fyzické architektuře jsou nicméně všechny tyto tabulky ukládány na stejné místo, což znamená, že ve fyzické architektuře se nachází pouze jedna vrstva.



Obrázek 7.4: Fyzická a logická architektura úložiště

V obou případech jsou z architektury vyňaty statistiky, které se ukládají na jiné místo, jedná se tedy o speciální vrstvu v architektuře.

7.4 Rozšíření typu ukládaných souborů

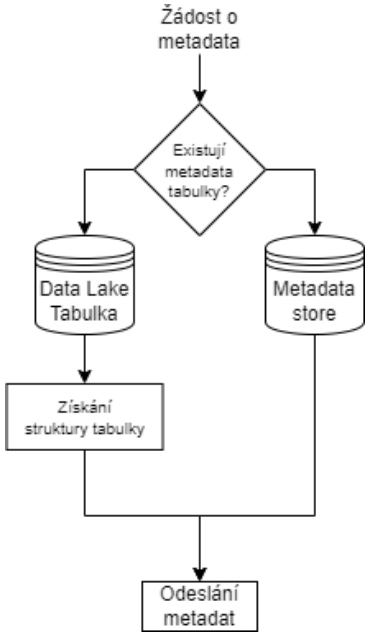
Datové jezero v tomto projektu bylo původně vytvořeno s medicinskymi daty. Tato data by výhradně ve formátu XML. Proto bylo při nahrávání dat do datového jezera počítáno pouze s tímto formátem dat. Data se obecně ale nenachází pouze ve formátu XML, proto byl projekt rozšířen o další možnosti ukládání. Mezi nejznámější a nejčastěji používané formáty dat patří JSON a CSV. Pokud nyní uživatel nahraje data v tomto formátu, program se rozhodne na základě přípony souboru, jakou metodu využít při čtení a ukládání dat. Původně se program rozhodoval o automatickém uložení dat do příslušné tabulky na základě root XML tagu. Protože JSON soubory nemají root tag, je tato možnost při nahrávání dat ignorována.

7.5 Alternativní názvy sloupců

Sloupce mohou mít při vytvoření názvy, které nemají pro koncového uživatele význam. Aby bylo možné s tabulkami lépe pracovat, bylo úložiště rozšířeno o možnost přidání vlastních názvů sloupců. Tuto možnost uživatel najde na stejné stránce pod tlačítkem “Správa názvu sloupců”. Po kliknutí na tlačítko se zobrazí modální okno, ve kterém může uživatel napsat libovolný název, který nahradí při vizualizaci současný název sloupce.

Po stistknutí tlačítka "Uložit"se názvy sloupců uloží do metadat ve složce `/upload/metadata/[název_tabulky]/[název_tabulky].json`. Z tohoto souboru jsou metadata o sloupcích později získávána jako první. Tyto názvy se zobrazují při

vizualizaci datové struktury tabulky, následně pak také při výběru sloupců k logistické regresi a při vizualizaci statistik. Postup pro získání schématu tabulky je vizualizován na obr. 7.5.



Obrázek 7.5: Proces nahrávání názvů sloupců pro vizualizaci

7.6 Vizualizace schémat tabulky

Aby bylo možné porozumět údajům v tabulce a pracovat s nimi dále, je potřeba znát její strukturu. Tato funkce doposud v aplikaci implementována nebyla. Nejbližší možností, jak vizualizovat tabulku, bylo použití příkazu "describe [název_tabulky]", který v plaintextu zobrazil metadata o tabulce. Tabulky se mohou v Delta lake však vyvíjet, což už nikde vizualizovatelné není. Možnosti pro vizualizaci těchto metadat jsou 2:

1. Využití dat z příkazu "describe", který vrátí informace o tabulce v Dataset objektu. Příkaz vrátí metadata tak, jak jsou zobrazována při vyvolání SQL příkazu v sekci "Vykonat SQL dotaz". Při úpravě vizualizace je toto možné, nicméně nastává problém s 2. potřebou - vizualizace vývoje tabulky. Pro tento případ byl vytvořen SQL příkaz "describe history [název_tabulky], který by měl tento problém vyřešit. Nicméně při testování této funkce byl nalezen problém. Tento příkaz není dostupný ve všech SQL databázích a MySQL databáze, která je využívána v tomto projektu, jej neimplementuje.

2. Extrakce metadat z lakehouse. `Delta Lake` si uchovává všechna metadata o vytvoření a úpravě jednotlivých tabulek ve formátu JSON. Složka, ve které se tato data nachází se jmenuje `lakehouse/[název_tabulky]/_delta_log`. Jednotlivé logy se nachází v souborech s názvem, který je složen z 20 číslic (př. 00000000000000000001.json). Tyto soubory se číslují od 0 a každý log obsahuje jednu z několika operací.

    - Přidání souboru - přidá datový soubor.
    - Odebrání souboru - odstraní datový soubor.
    - Aktualizace metadat - aktualizuje metadata tabulky (např. změní název, schéma nebo rozdělení tabulky).
    - Nastavení transakce - Zaznamená, že strukturovaná úloha streamingu odevzdala mikrodávku s daným ID.
    - Změna protokolu - Umožňuje nové funkce přepnutím transakčního protokolu Delta Lake na nejnovější softwarový protokol.
    - Informace o revizi - Obsahuje informace kolem revize, která operace byla provedena, odkud a v jakém čase.

Každý z těchto souborů má k sobě přiložený i CRC soubor (např. 00000000000000000001.json.crc). CRC je zkratka pro Cyclic redundancy check. Jedná se o kód sloužící k opravě chyb dat. V případě, že by byl některý z logů poškozen, stává se log nepoužitelným a není možné se v případě potřeby vrátit do předchozího stavu tabulky, neboť logy se používají k rekreaci tabulky v určitém stavu. Díky CRC souboru je možné zkontrolovat, zdali nejsou v log souboru chyby, které se tam mohly vyskytnout při ukládání souboru na disk. V případě, že je chyba nalezena, může být díky CRC opravena a předejdeme tedy chybě. Tyto soubory nejsou při implementaci k ničemu použity, nicméně je vhodné o jejich existenci vědět. Bez této vědomosti by se mohl programátor rozhodnout nahrát všechny soubory ve složce, data v nahraných souborech by pak nedávala smysl.

K vizualizaci jsou potřeba pouze JSON soubory. Na základě požadavku uživatele k vizualizaci struktury tabulky jsou jednotlivé JSON soubory načteny aplikací. Tyto soubory je třeba následně filtrovat podle toho, jaký typ metadat soubor obsahuje.

Soubory nejsou podle standardu validní JSON soubory. Každý soubor obsahuje podle typu logu několik JSON objektů, které však nejsou nijak spojeny do jednoho validního souboru. Namísto toho se v souboru nachází objekty, které nejsou oddělené čárkou, ani spojeny do pole.

Jednotlivé JSON objekty jsou ze soubory parsovány a podle typu obsažených metadat jsou vyfiltrovány. Data ze souborů obsahující metadata se změnami struktury tabulky jsou následně odeslány k vizualizaci na straně webového klienta.

Vizualizace metadat tabulky se nachází na obrazovce úpravy tabulky. Stránka je obohacena o Preact vizualizaci. Na stránce jsou zobrazena 2 okna. První okno zobrazuje současnou datovou strukturu tabulky. Struktura tabulky je vypisována ve tvaru `[název_sloupce]: [datový_typ]`. Pokud se jedná o datový typ pole, jsou okolo datového typu obsaženého v poli závorky [] značící, že se jedná o pole. V případě, že je obsahem sloupce datová struktura, místo datového typu je vykreslí znak "+". Poté, co na řádek uživatel klikne, rozbalí se obsah datové struktury. Takto se data vykreslují rekurzivně.

Obrázek 7.6: Vizualizace schématu tabulky

Ve druhém okně je možné zkoumat změny v datové struktuře tabulky. Díky atributu "timestamp" nacházejícím se v JSON objektu zaslaným s metadaty je možné vybrat porovnání mezi strukturou ve 2 rozdílných datech. Je možné vybrat libovolné 2 datumy a porovnat je.

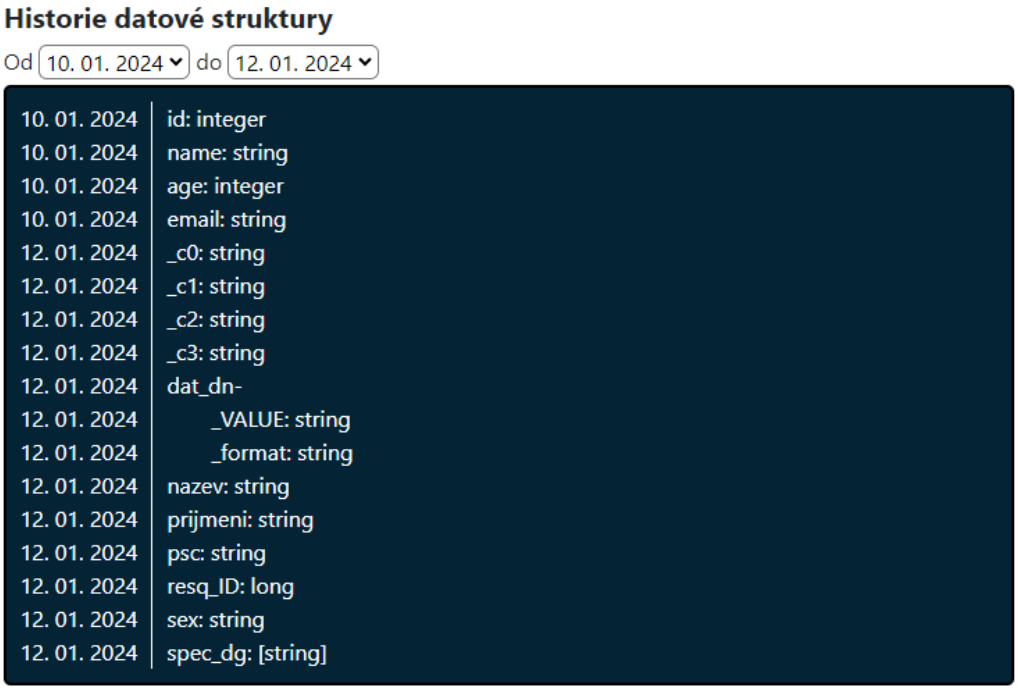
U atributu se také vykresluje, kdy byl přidán. Na obrázcích 7.6 a 7.7 se nachází komponenty pro vizualizaci schématu tabulky a jeho historického vývoje.

7.7 Výběr statistických metod

Statistiky jsou pro datovou analýzu důležité zejména pro:

- Ověřování správného nahrávání dat do úložiště. Do úložiště se mohou dostat nesprávné hodnoty, které by se v jednotlivých atributech neměly nacházet, při kontrole statistik jednotlivých tabulek je možné tyto hodnoty kontrolovat.
- Tvorbu predikcí a další statistické metody. Díky statistikám, např. maximálním, minimálním a průměrným hodnotám v číselných sloupcích je možné zkontrolovat, zdali se nachází ve sloupci extrémní hodnoty. Tyto informace opět slouží uživateli ke kontrole dat.

Pro zobrazování těchto dat byla použita stránka, na které se nachází komponenta pro logistickou regresi. Toto řešení bylo zvoleno z důvodu, že uživatel pravdě-



Obrázek 7.7: Komponenta umožňující porovnání schémat ve vybraných datech

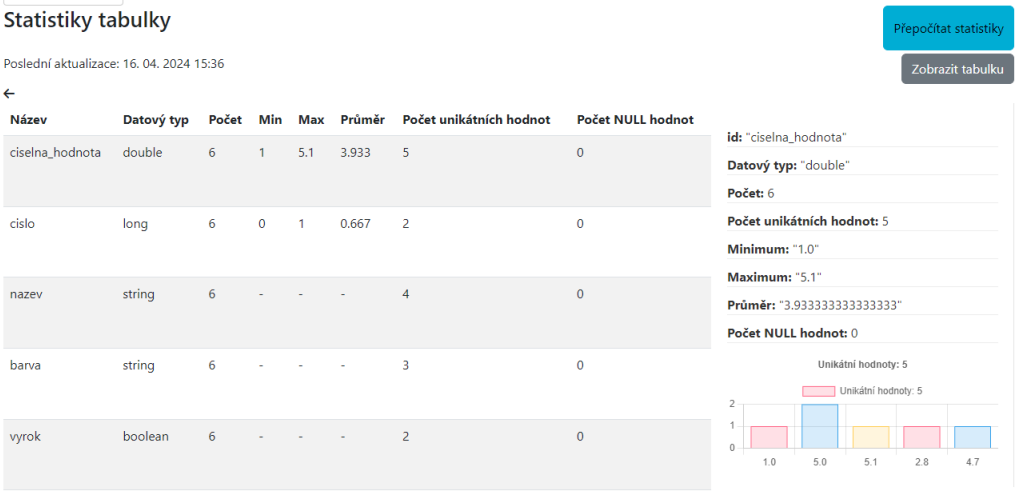
podobně bude nejdříve data zkoumat, až následně bude chtít ověřit svoji hypotézu pomocí analytické funkce. Spojením těchto funkcí do jedné stránky se eliminuje nutnost přepínat mezi případnými dvěma stránkami, nebo nutnost mít vedle sebe otevřená 2 okna.

Uživatel kliknutím na položku v rozbalovacím okně vybere tabulku, jejíž data chce zkoumat. Aplikace odešle požadavek serveru, jenž získá z JSON souboru požadované statistiky a odešle je zpět do aplikace. Po získání požadovaných dat se zobrazí komponenta statistik a vykreslí se tabulka s jednotlivými statistikami. Mezi tyto statistiky patří:

- Název sloupce
- Datový typ
- Počet řádků
- Minimum
- Maximum
- Průměr
- Počet unikátních hodnot

- Počet nullových hodnot

Statistiky minimum, maximum a průměr jsou dostupné pouze pro číselné hodnoty. Tuto tabulku a statistiky je možné dále jednoduše rozšiřovat pouhým přidáním statistik v serverové části aplikace a přidáním statistiky do pole, které určuje vykreslení jednotlivých sloupců.



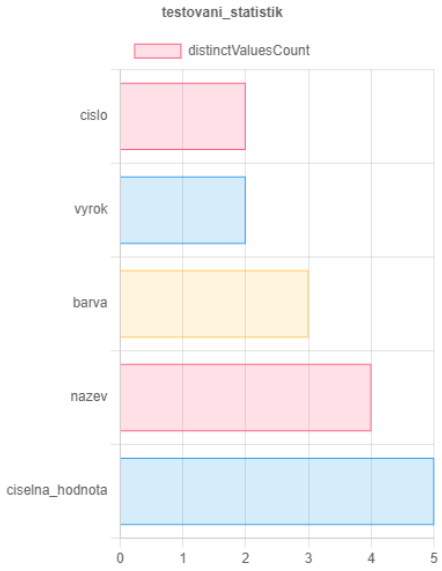
Obrázek 7.8: Vizualizace tabulky statistik

Tabulky je možné řadit podle hodnot v jednotlivých sloupcích kliknutím na hlavičku vybraného sloupce jednou, nebo dvakrát. Podle počtu kliknutí se tabulka seřadí vzestupně, či sestupně.

Kliknutím na libovolný řádek se vpravo od tabulky zobrazí detail řádku. Zobrazí se stejné hodnoty, které jsou v řádku tabulky, navíc se však zobrazí graf s unikátními hodnotami obsaženými ve sloupci a počet jejich výskytů. V případě, že je unikátních hodnot více než 6, zobrazí se místo grafu tabulka. Toto řešení bylo zvoleno z důvodu čitelnosti, grafy s větším množstvím sloupců v detailu řádky se stávají velice rychle nepřehlednými.

Detail vpravo od tabulky se tlačítkem "`Zobrazit tabulku`" dá změnit do módu tabulky. Kliknutím na hlavičku tabulky se pak zobrazí graf, ve kterém budou zobrazeny hodnoty obsažené v daném sloupci. Je možné zobrazit pouze hodnoty v číselných sloupcích a sloupci s datovým typem, kde bude zobrazen součet výskytů jednotlivých datových typů.

Statistiky je možné si na žádost nechat okamžitě přepočítat. Obrázky 7.8 a 7.9 ukazují vzhled implementovaných tabulek a grafů.



Obrázek 7.9: Vizualizace četností unikátních hodnot v jednotlivých sloupcích

7.8 Dostupné knihovny pro datovou analýzu

Knihoven s analytickými funkcemi existuje pro Javu celá řada, není potřeba programovat celé analytické funkce od začátku. V následující kapitole budou popsány a porovnány vybrané dostupné knihovny a vybrána nejvhodnější pro použití v projektu. Podpora programovacího jazyka Java byla hlavním faktorem pro výběr knihoven.

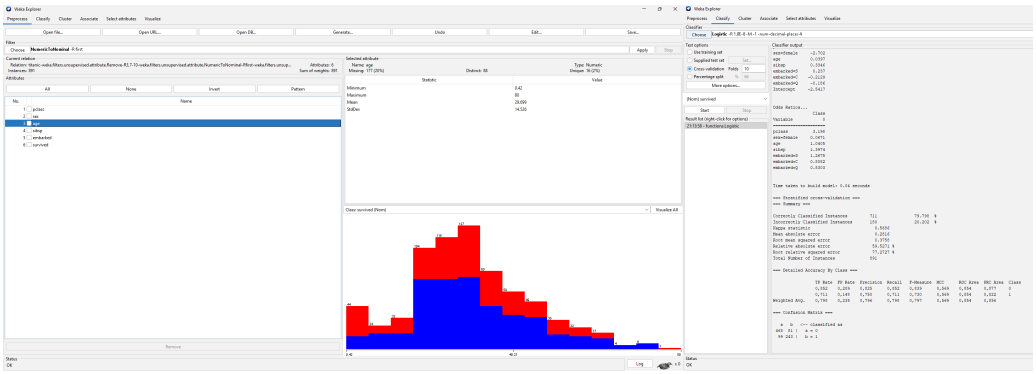
7.8.0.1 Weka

Weka je jednou z nejstarších a nejpoužívanějších knihoven pro strojové učení v jazyce Java. Má otevřený zdrojový kód a lze ji používat zdarma.

Weka (Waikato Environment for Knowledge Analysis) zahrnuje řadu nástrojů pro různé úlohy včetně klasifikace dat, regrese, dolování asociačních pravidel a shlukování. Díky intuitivnímu grafickému rozhraní usnadňuje Weka předzpracování dat, vizualizaci a vyhodnocování modelů jak začátečníkům, tak zkušeným uživatelům.

Program Weka lze používat prostřednictvím rozhraní Java API, standardních terminálových aplikací nebo grafického uživatelského rozhraní [21].

Výhodou knihovny Weka je právě existující aplikace pro PC, která disponuje uživatelským rozhraním. Aplikace je jednoduchá na použití a poskytuje všechny implementované metody pro datovou analýzu.



Obrázek 7.10: Grafické uživatelské rozhraní aplikace Weka

7.8.0.2 JSAT

JSAT je knihovna pro podporující klasifikaci dat, regrese, shlukování a základní metody strojového učení. JSAT nemá žádné externí závislosti a je čistě v Javě. Téměř všechny algoritmy jsou nezávisle implementovány pomocí objektově orientovaného rámce. JSAT je k dispozici pod licencí GNU GPL.

Autor knihovny ve svém porovnání tvrdí, že JSAT je rychlejší, než Weka [22].

Knihovna je nicméně tvořena pouze jedním člověkem, což by mohlo v případě nalezení chyby znamenat dlouhou dobu opravy.

7.8.0.3 Apache Commons Math

Apache Commons Math představuje rozsáhlou sbírku matematických a statistických utilit pro programovací jazyk Java, které jsou navrženy tak, aby byly snadno integrovatelné do aplikací a umožňovaly pokročilou analýzu. Tato knihovna poskytuje rozmanité nástroje v oblastech jako je numerická algebra, diferenciální geometrie, konstrukce a trénování neuronových sítí, operace s maticemi, a aplikace strojového učení. Přestože dokumentace obsahuje řadu užitečných příkladů použití, spektrum funkcionalit, které Apache Commons Math nabízí, je mnohem širší a umožňuje řešení komplexních výpočetních úkolů [23].

Výhodou knihovny Apache Commons Math je to, že se jedná o další z knihoven ze sady Apache, má tedy velikou podporu ze strany mnoha vývojářů. Dokumentace, přestože je kompletní, se nejeví jako uživatelsky přívětivá a chybí v ní příklady použití.

7.8.1 Extrakce dat pro datovou analýzu

Data, která jsou uložena v datovém jezeře nejsou úplně kompatibilní s metodou logistické regrese. Data mohou být různě zanořena do dalších struktur, nebo polí,

což není pro datové analýzy vhodné z důvodu ztráty významu jednotlivých dat uvnitř datových struktur. Proto bylo potřeba v těchto případech potřeba upravit vstupní data.

Data mohou nabývat díky nekonečnému zanořování datových struktur do sebe nekonečně mnoha dimenzí, logistická regrese nicméně podporuje pouze data ve formě 2 dimenzí, jinak řečeno standardní tabulky se sloupci a řádky.

Jednou z možností, jak tento problém vyřešit, je každému objektu přiřadit hash, nebo jakékoliv číslo, které bude tento objekt reprezentovat. Pokud se daný objekt bude nacházet ve více řádcích, bude v daných řádcích nabývat stejné hodnoty. Tento přístup je nejjednodušší, nicméně problémem této metody je naprostá ztráta významu původních dat v analýze.

Pro uchování významu dat je potřeba převést tato data do formy použitelné v logistické regresi. Toho bylo docíleno tak, že v případě, že sloupec obsahuje jako datový typ objekt, jsou jeho atributy převedeny jako sloupce do tabulky s prefixem názvu sloupce, ve kterém se objekt nachází. Takto se postupuje rekurzivně i v případě, že objekt obsahuje jako atribut další objekt. Díky tomu se eliminují nadbytečné dimenze.

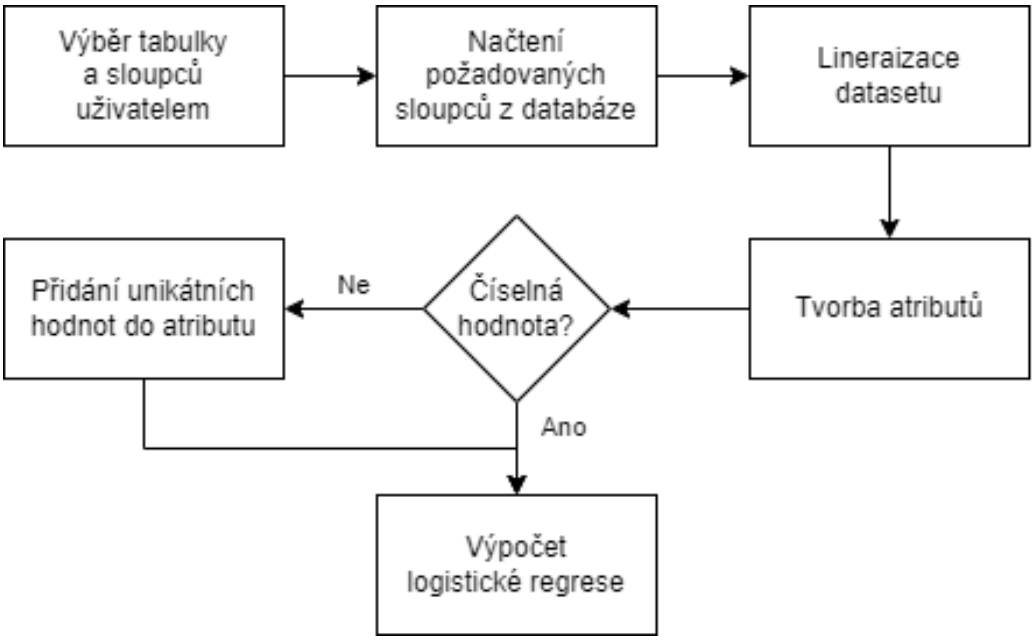
Podobně se musí přistupovat k polím dat ve sloupci. Data v poli se musí převést do jedné významové hodnoty, kterou je možné vložit k analýze logistické regresi. Možností je několik, přičemž 2 z nich zde již byly zmíněny. Nahrazení pole za jednu hodnotu reprezentující dané pole je znovu možností, při které dochází ke ztrátě významu hodnot v poli. Dále je možné převést jednotlivé hodnoty v poli na sloupce v tabulce. Tato metoda je nicméně nevhodná z důvodu, že v jednotlivých záznamech mohou pole mít různé délky, což by vedlo k tomu, že dlouhá pole by měla za dopad vytvoření mnoha sloupců a v případě, že se jedná o výjimku, by zbytek hodnot byl prázdný, což by ovlivnilo výsledek regrese.

Podobným způsobem by se daly hodnoty rozdělit do nových řádků v závislosti na počtu hodnot v poli. Tím by vznikly nové řádky s opakující se hodnotou ve sloupcích v nominální hodnotou a jednotlivými hodnotami z pole. Tato metoda je nicméně pro výzkum dat nedostačující. Problémem tohoto přístupu je to, že pro výzkum dat pomocí regrese se předpokládá nezávislost jednotlivých pozorování. Toto pravidlo je porušeno roznásobením řádků s poli, a proto by výsledek regrese nebylo možné považovat za vypovídající.

Další možností je vytvoření hodnoty na základě podobnosti polí. Tato metoda je nicméně velice specifická a je potřeba mít daný use case dobře prozkoumaný.

Další možností pro pole číselných hodnot je délka vektoru, což má opět velice úzké využití. Poslední, zvolenou možností je vytvoření hodnoty na základě statistiky vybrané uživatelem. Uživateli je umožněno na základě typu hodnot uložených v poli vybrat, podle čeho bude hodnota vypočítána. U řetězcových hodnot byly jako vhodné metody vybrány unikátní hodnoty v poli a celková délka pole, u číselných

hodnot byly možnosti rozšířeny ještě o maximum, minimum, průměr, medián a součet. Uživatel si tedy na základě svého use-case vybere požadovanou statistiku.



Obrázek 7.11: Extrakce dat pro datovou analýzu

7.8.2 Implementace logistické regrese

Uživatel, který chce provádět analýzu, nejdříve vybere tabulku, nad kterou chce analýzu provést. Poté, co se načtou sloupce z databáze, je mu umožněno vybrat sloupce, které chce zahrnout do analýzy. Pokud je nějaký ze sloupců typu pole, musí si vybrat, jak data nahradit, neboť, jak již bylo zmíněno v předchozí kapitole, logistická regrese neumožňuje jako vstup pole. Poté, co uživatel vybere všechny požadované sloupce, klikem na jeden ze sloupců vybere, který z nich bude označen jako třída k analýze. Až poté může kliknout na tlačítko "`Analyzovat`".

Poté, co server obdrží žádost k analýze, je načtena celá tabulka. K načtení celé tabulky bylo přistoupeno z důvodu rychlejšího načítání. Porovnány byly 2 možnosti načítání, jedna z nich pomocí metody "`load`" a 2. pomocí vykonání SQL příkazu. Výsledkem porovnání bylo, že načítání dat příkazem load je instantní, zatímco načítání dat SQL příkazem je pomalejší. Návratovou hodnotou je objekt "`Dataset<Row>`". `Dataset` je datová struktura ve `SparkSQL`, která je silně typovaná a je mapou na relační schéma. Představuje strukturované dotazy s kódovači. Jedná se o rozšíření API datového rámce. `Spark Dataset` poskytuje typovou bezpečnost i objektově orientované programové rozhraní [24].

Dataset obsahuje nejen řádky získané z databáze, obsahuje i kompletní datové schéma. Díky tomu je možné s daty nadále typově pracovat a data transformovat.

Datový typ je možné získat příkazem **dataset.schema()**. Tímto příkazem je získáno celé datové schéma, pro zpracování je potřeba získat seznam polí. Toho je docíleno příkazem **schema.fields()**. Návratovou hodnotou této funkce je pole **StructField**. StructField je objekt, z nějž jsou důležité 3 položky: **name, data type a element type**. Položka name je důležitá z důvodu filtrace vybraných polí od uživatele. Data type je atribut obsahující informaci o datovém typu sloupce.

DataTypes je třída obalující enum všech možných datových typů ve SparkSQL. Dostupnými datovými typy jsou:

- **StringType** - textové řetězce
- **BinaryType** - binární pole
- **BooleanType** - hodnota true nebo false
- **DateType** - obalová hodnota pro Java Date
- **TimestampType** - unixový čas
- **CalendarIntervalType** - délka časového intervalu vyjádřená v měsících, dnech a milisekundách
- **DoubleType**, **FloatType**, **ByteType**, **IntegerType**, **LongType**, **ShortType** - číselné hodnoty
- **NullType**

V závislosti na datovém typu ve sloupci je možno dále s hodnotami pracovat.

Následuje transformace dat do formátu **Instances**, který je vyžadován knihovnou Weka. K tomu je potřeba schématu získaného v předchozím kroku. Díky tomu se vytvoří seznam **Attributes**, který představuje jednotlivé sloupce, položky v objektu **Instances**.

Attribute podporuje následující typy:

- **Číselný** - Tento typ atributu představuje číslo s pohyblivou řádovou čárkou.
- **Nominální** - Tento typ atributu představuje pevnou množinu jmenovitých hodnot.
- **String** - Tento typ atributu představuje dynamicky se rozšiřující množinu jmenovitých hodnot. Obvykle se používá při klasifikaci textu.

- **Date** - Tento typ atributu představuje datum, interně reprezentované jako číslo s pohyblivou řádovou čárkou, které uchovává milisekundy od 1. ledna 1970, 00:00:00 GMT. Řetězcová reprezentace data musí být v souladu s ISO-8601, výchozí je yyyy-MM-dd'T'HH:mm:ss.
- **Relační** - Tento typ atributu může obsahovat další atributy a používá se např. pro reprezentaci údajů Multi-Instance. (Data Multi-Instance se skládají z nominálního atributu obsahujícího bag-id, dále z relačního atributu se všemi atributy sáčku a nakonec z atributu třídy) [25].

Pro případ využití dat v logistické regresi se využívají pouze atributy číselné a nominální.

Testování 8

Aplikace byla testována uživatelsky. Pro ověření funkčnosti byly vytvořeny testovací scénáře, které mají za cíl pokrýt ověření správné funkčnosti všech nově implementovaných funkcionalit. Testovací scénáře všechny uvažují správné spuštění aplikace.

8.1 Vytvoření nové tabulky

Do aplikace byla přidána možnost vytvoření nových tabulek podle schémat dodaných ve formátu CSV a JSON. Cílem je ověřit správné vytvoření tabulek a vložení dat.

Vstupní podmínky

1. 1 soubor se schématem ve formátu CSV, 1 soubor se schématem ve formátu JSON.
2. Data ve formátu JSON a CSV odpovídající schématu tabulky.
3. Data neodpovídající schématu.

Postup

1. Vložení souboru do Data Lakehouse.
2. Vložení datového souboru s odpovídající strukturou.
3. Vložení datového souboru s rozdílnou strukturou.

Očekávané výsledky

1. Vytvoření tabulky s odpovídajícím názvem.
2. Přijetí souboru s odpovídající strukturou.
3. Odmítnutí souboru s rozdílnou strukturou. Uložení souboru do úložiště pro neodpovídající data.

Výsledky
Všechny očekávané výsledky byly naplněny.

8.2 Alternativní pojmenování sloupců

Alternativní názvy sloupců jsou další implementovanou funkcí. Alternativní názvy by se měly zobrazovat uživateli vždy, pokud je alternativní název dostupný.

Vstupní podmínky

1. 1 vytvořená tabulka.

Postup

1. Vytvoření alternativních názvů ve formuláři pro jejich tvorbu.

Očekávané výsledky
Alternativní názvy sloupců se mají zobrazit v:

1. Vizualizaci struktury tabulky a její historii,
2. Odvozování nových tabulek,
3. Vizualizaci statistik,
4. Výběru sloupců pro logistickou regresi.

Výsledky
Všechny očekávané výsledky byly naplněny.

8.3 Logistická regrese

Cílem tohoto testu je ověřit správnost transformace dat pro logistickou regresi a ověření výsledků metody. Vzhledem k tomu, že implemetace logistické regrese proběhla s knihovnou Weka, je potřeba výsledky regrese porovnat s výslekdy uvnitř programu Weka. Aplikace nedovoluje multinominální regresi, test by tedy měl ověřit, že program zobrazí chybovou hlášku, pokud bude ke klasifikaci vybrán sloupec s více než 2 unikátními hodnotami.

Vstupní podmínky

1. 1 vytvořená tabulka s více než 10 záznamy,
2. alespoň 1 sloupec s řetězci a více než 2 unikátními hodnotami,
3. alespoň 1 sloupec s řetězci a právě 2 hodnotami,

4. alespoň 1 sloupec s booleovskými hodnotami,
5. alespoň 1 sloupec číselný a více než 2 unikátními hodnotami,
6. alespoň 1 sloupec číselný a právě 2 hodnotami,
7. alespoň 1 sloupec s polem čísel,
8. alespoň 1 sloupec s polem řetězců,
9. libovolný sloupec s právě 1 unikátní hodnotou.

Postup

1. Výběr tabulky pro logistickou regresi.
2. Výběr všech sloupců v různých permutacích.
3. Postupný výběr všech sloupců pro klasifikaci.
4. Porovnání výsledků s výsledky v aplikaci Weka.

Očekávané výsledky

1. Chybová hláška pro sloupce, kde se nachází více než 2 unikátní hodnoty.
2. Stejné výsledky pro případy, kdy byl ke klasifikaci vybrán sloupec se 2 unikátními hodnotami.
3. Chybová hláška pro sloupce, které byly přidány pro analýzu a kde se nachází pouze 1 unikátní hodnota.

Výsledky
Všechny očekávané výsledky byly naplněny.

8.4 Vytvoření odvozené tabulky

Při vytváření tabulky odvozené se postupuje odlišně, než při standardním zakládání tabulky ze souboru. V tomto scénáři je ověřena funkčnost vytvoření tabulky a její aktualizace na základě vstupu dat do podkladových tabulek. Odvozená tabulka se může vytvořit z 1 a více jiných tabulek, je proto potřeba otestovat všechny případy.

Vstupní podmínky

- 2 existující tabulky, které mají alespoň 1 společný sloupec sloužící ke spojení
- Alespoň 1 záznam v každé tabulce

Postup
Odvozená tabulka z 1 tabulky:

1. Výběr 1 tabulky pro odvození,
2. Výběr libovolných sloupců,

Odvozená tabulka z 2 tabulek:

1. Výběr 2 tabulek pro odvození,
2. Výběr libovolných sloupců,
3. Výběr sloupců pro propojení,
4. Alternativní průběh: Nevýběr žádného sloupce pro propojení,
5. Výběr typu spojení,

Společná část testu:

1. Alternativní průběh: Výběr žádného sloupce.
2. Přidání podmínky pro libovolný sloupec,
3. Nevložení žádné podmínky,
4. Vložení dat do podkladové tabulky,

Očekávané výsledky

1. Chybová hláška, pokud není vybrán žádný sloupec.
2. Vytvoření nové tabulky s naplněnými daty, pokud jsou všechny podmínky splněny.
3. Chybová hláška, pokud není vybrán sloupec k propojení 2 tabulek.
4. Aktualizace statistik odvozené tabulky po vložení dat do podkladové tabulky.

Výsledky
Všechny očekávané výsledky byly naplněny.

8.5 Výpočet statistik

Statistiky jsou jednou z nejdůležitějších funkcionalit implementovaných v této práci. Data ze statistik by měla být přesná a aktualizovaná po přidání nových dat do tabulky. Také by mělo jít vždy ručně aktualizovat statistiky pro případ přidání nového záznamu do tabulky skr SQL rozhraní. Cílem je ověřit správnost výsledků statistik, jejich automatickou aktualizaci a ručně vynucenou aktualizaci.

Vstupní podmínky

1. 1 vytvořená tabulka s více než 10 záznamy,
2. alespoň 1 sloupec s řetězci,
3. alespoň 1 sloupec s booleovskými hodnotami,
4. alespoň 1 sloupec číselný,
5. 1 sloupec s vnořenou datovou strukturou,

Postup

1. Vložení nového záznamu do tabulky z datového souboru,
2. Ověření aktualizace všech statistik tabulky,
3. Vložení nového záznamu s pomocí SQL,
4. Stisknutí tlačítka "`Přepočítat statistiky`"na stránce "`Datová analýza`",
5. Ověření aktualizace statistik.

Očekávané výsledky

1. Statistiky by se měly aktualizovat automaticky po vložení záznamu z datového souboru.
2. Statistiky by se měly přepočítat po stitknutí tlačítka "`Přepočítat statistiky`".

Výsledky

Všechny očekávané výsledky byly naplněny.

Informační systém Medical Research and Education

9

9.1 Popis systému MRE

`Medical Research and Education` [26] je platforma pro navrhování a vývoj informačních systémů pro výzkumné účely, zejména s lékařskými daty.

Cílem je dlouhodobá archivace a opakované použití s důrazem na ochranu a bezpečnost dat.

Výzkumná skupina Medicínské informační systémy je součástí oboru Lékařská informatika na Katedře informatiky a inženýrství Fakulty aplikovaných věd Západočeské univerzity v Plzni. Výzkumná skupina je součástí výzkumného centra NTIS - Nové technologie pro informační společnost.

Existuje několik modulů, z nichž důležité pro tuto práci jsou následující:

1. AnonMed je vhodný pro deidentifikaci obsahu metadat v několika formátech souborů.

2. PureImage identifikuje veškerý textový obsah v pixelových datech a provádí jeho klasifikaci. Je také možné redigovat části obrázků s klasifikovaným textovým obsahem.

Oba nástroje mohou deidentifikovat jakoukoli identitu osoby, zařízení, zdravotnického zařízení nebo personálu.

9.1.0.1 Schéma metadat

Metadata mají své schéma v souladu s RDF a ontologiemi, včetně jazyka webové ontologie (OWL2).

9.1.0.2 Nová data

Nová data RDF může vytvářet webová aplikace HTML5, generátor webových formulářů.

9.2 Srovnání Data Lakehouse a MRE

Jedním ze stanovených cílů práce je porovnat získané výsledky s aplikací `MRE`. Aplikace `MRE` se sestává z mnoha modulů, jedná se o aplikaci, která je vyvíjena již více než 14 let a skládá se ze 17 modulů. Rozsah této práce nemůže pokrýt celý rozsah systému MRE.

Přesto má aplikace `Data Lakehouse` některé funkce, které poskytuje i systém MRE, `Data Lakehouse` navíc nyní poskytuje analytické rozhraní pro uživatele uvnitř aplikace.

9.2.1 Výhody Data Lakehouse

1. **Obecnost** - aplikace MRE umožňuje práci pouze s medicínskými daty, pro něž má připravenou strukturu ve své definici RDF. Aplikace Data Lakehouse však umožňuje práci i jinými daty, na obecnost aplikace byl kladen velký důraz při rozšiřování funkcionalit. Uživatel může nyní nahrát libovolné schéma tabulky do Data Lakehouse a okamžitě do ní začít nahrávat data, uživatel není limitován RDF předpisem. Díky tomu má Data Lakehouse mnohem širší využití, než pouze medicínská data.

2. **Datová analýza přímo v aplikaci** - Data Lakehouse byl rozšířen o vrstvu pro datovou analýzu a statistiku. Systém MRE těmito možnostmi do jisté míry také disponuje, uživatel však musí data z MRE nejdříve ručně extrahovat, tato data pak následně nahraje do systému, který je pro datovou analýzu určen. Tento krok byl v Data Lakehouse eliminován, analýza se může provádět přímo v aplikaci, což uživateli zjednoduší práci.

3. **Tvorba dotazů** - V případě, že chce uživatel extrahovat data, musí se k tomu použít jazyk SQL, v případě MRE se jedná o SPARQL. Jazyk SQL je obecně známější a využívanější, než SPARQL, díky čemuž je větší pravděpodobnost, že uživatel dokáže vytvořit vlastní dotaz.

9.2.2 Současné nedostatky

Aplikace má v porovnání s MRE i mezery, které by do budoucna bylo vhodné eliminovat. Tyto nedostatky jsou brány v potaz a jsou zahrnuty i do kapitoly 10, která pojednává o budoucím vývoji aplikace.

- Absence nahrávání dalších typů souborů - Aplikace MRE uchovává ve svém úložišti obrazové záznamy DICOM. Digital Imaging and Communications in Medicine je mezinárodní standard pro lékařské snímky a související informace. Definuje formáty pro lékařské snímky, které lze vyměňovat s daty a v kvalitě nezbytné pro klinické použití [16]. Vizualizace, nebo analýza těchto záznamů by mohla přinést významné výsledky.

9.2.3 Shrnutí porovnání

Aplikace `Data Lakehouse` poskytuje nyní uživateli uživatelské rozhraní, ve kterém může sám data zkoumat a testovat. Byl také vytvořen prototyp pro výpočet statistik a jejich následné zobrazení. S aplikací se pracuje snáze, provedení dotazu v jazyce SQL, nebo použití formulářů v aplikaci je snazší, než provedení dotazu v aplikaci `MRE` s pomocí SPARQL. Byl položen základ pro další rozšíření této platformy v oblasti datové analýzy. Přestože aplikace `Data Lakehouse` nedisponuje možností ukládání obrazových záznamů, je možné tuto funkcionalitu v budoucnu možné implementovat a eliminovat tak jeden z nedostatků této aplikace v porovnání s `MRE`.

Návrhy na zlepšení 10

10.1 Vytvoření datového slovníku

Datový slovník je soubor názvů, definic a atributů datových prvků, které se používají nebo zachycují v databázi, informačním systému nebo v rámci výzkumného projektu. Popisuje významy a účely datových prvků v kontextu projektu a poskytuje pokyny pro interpretaci, přijaté významy a reprezentaci. Datový slovník rovněž poskytuje metadata o datových prvcích. Metadata obsažená v datovém slovníku mohou pomoci při definování rozsahu a charakteristik datových prvků, jakož i pravidel pro jejich použití a aplikaci [27].

V kombinaci se statistikami by se datový slovník mohl použít pro hledání slov v názvu tabulek, sloupců, nebo jejich obsahu. Díky datovému slovníku bude možné jednodušeji vytvářet odvozené tabulky, nebo hledat vhodná vstupní data pro datovou analýzu pomocí analytických metod.

10.2 Přidání dalších analytických metod

Jako nejdůležitější analytická metoda pro zkoumání medicínských dat byla určena logistická regrese. Pomocí této metody se hledají korelace mezi jednotlivými atributy. Pro různá další data logistická regrese nemusí být nejlepší volbou, proto by bylo dobré implementovat další analytické metody.

10.3 Rozšíření množství statistik

V práci bylo vybráno a ukládáno několik vybraných statistik. Pro uživatele mohou být vhodné nicméně i další, v této práci nezahrnuté statistiky.

10.4 Batchové zpracování statistik

V současné chvíli se statistiky tabulek zpracovávají po každém přidaném záznamu do tabulky. To může pro server znamenat velkou zátěž v případě, že bude do databáze nahráno velké množství záznamů najednou. Proto by bylo vhodnější zpracovávat statistiky batchově v pravidelných časových intervalech, nebo v případech, kdy se po určitou dobu do databáze nic nenahrávalo. Využít by se mohl například balíček Spring Batch.

10.5 Úprava pipeline pro odvozené tabulky

Tabulky, které jsou nyní vytvářeny odvozením od původních tabulek nemají aktuálně možnost úpravy vstupních parametrů, což má za následek to, že v případě, že chce uživatel do odvozené tabulky přidat další sloupec, je nucen namísto úpravy parametrů pro odvozování tabulek existující tabulku smazat (nebo si ji ponechat) a vytvořit tabulku novou. Toto bude pravděpodobně pro uživatele nepříjemné, proto je potřeba umožnit tuto funkcionalitu.

10.6 Ukládání dalších nestrukturovaných dat

V současné chvíli Data Lakehouse umožňuje uživateli nahrávat pouze záznamy ve formátu JSON, XML, nebo CSV. Obzvlášť pro medicínská data uvedená v této práci je nicméně důležité ukládat i DICOM obrazové záznamy. Cílem je tedy umožnit uživateli přidávat nestrukturovaná data (libovolné obrázky, videa, a další) a mít možnost tato data propojit s vybraným záznamem v libovolné tabulce.

Závěr 11

Cílem práce bylo seznámit se s konceptem úložiště Data Lakehouse. Byla prozkoumán koncept datových jezer a skladišť, z nich vznikají koncepty typu lakehouse. Následně byly popsány a porovnány typy datových pump ETL a ELT. Zkoumán byl také formát úložiště Parquet, který se v úložištích využívá.

Poté byla zmapována dostupná data. Data se týkají oblasti medicínských dat a zkoumána byla jejich struktura a standardy, podle kterých se soubory ukládající data vytváří. Analyzována byla také platforma MRE, která s tímto typem dat pracuje.

Následně byla zkoumána aplikace Data Lakehouse, která byla vytvořena v předchozí práci. Byly zmapovány dostupné funkce a práce s aplikací. Byly zjištěny nedostatky aplikace a na nich byly v pozdější kapitole stanoveny funkcionality, které v aplikaci chybí a které je potřeba implementovat. V aplikaci dosud chyběla možnost jakékoliv analýzy dat, výpočet statistik, vizualizace datových struktur a možnost dalšího čištění dat pomocí datových pump.

O definované nedostatky byla následně aplikace rozšířena. Využita byla stávající aplikace, rošíření pokračovalo ve frameworku Spring Boot, do aplikace byly přidány knihovny Preact.js a Chart.js, které umožňují dynamickou práci s webovou aplikací a zlepšují interakci s uživatelem díky redukci množství přenačítání webové aplikace.

Aplikace byla následně funkčně a uživatelsky testována. Byly stanoveny testovací scénáře, podle kterých se při testování postupovalo. Díky testům se odstranily chyby při implementaci a vylepšila vizualizace, některých implementovaných funkcionalit.

Aplikace v porovnání s platformou MRE nyní umožňuje výpočet statistik pro libovolné tabulky nacházející se v úložišti, možné je také nově vytvářet odvozené tabulky, které mají sloužit pro následnou analýzu díky předchozímu čištění dat pomocí datové pumpy, která byla také implementována. Uživatel také může zkoumat statistiky jednotlivých tabulek a na jejich základě tvořit analýzy, což dosud možné nebylo, aplikace byla rozšířena o datovou analýzu a analytické zpracování dat.

Na závěr byly stanoveny některé funkcionality, o které by bylo možné Data Lakehouse rozšířit v budoucnu.

Všechny cíle diplomové práce byly splněny.

Seznam zkratek

- MRE - Medical Research and Education
- NoSQL - Not only SQL
- SQL - Structured Query Language
- BI - Business Intelligence
- CSV - Comma Separated Value
- JSON - JavaScript Object Notation
- OLAP - On-line analytical processing
- OLTP - On-line transaction processing
- ETL/ELT - Extract, transform, load / Extract, load, transform
- API - Application Programming Interface
- ODBC - Open Database Connectivity
- JDBC - Java Database Connectivity
- AWS - Amazon Web Services
- MVC - Model-View-Controller
- HTML - Hypertext Markup Language
- JVM - Java Virtual Machine
- REST - Representational State Transfer
- XML/XSD - Extended Markup Language / XML Schema Definition
- ERA - Entity-relationship model

Uživatelská dokumentace

B

Tato práce rozšiřuje již existující aplikaci Data Lakehouse, z toho důvodu zde budou popsány pouze části týkající se rozšíření. Pro celkové pochopení aplikace je nutné si přečíst uživatelskou dokumentaci přiloženou k práci `Koncept Data Lakehouse pro zpracování medicínských dat` [6].

B.1 Sestavení aplikace

Celý vývoj aplikace probíhal na operačním systému Windows 11 ve vývojovém prostředí IntelliJ IDEA. Výsledkem je Java aplikace Využívá Spring Boot framework, který má integrovaný servlet kontejner Tomcat, čímž odpadá nutnost konfigurace webového serveru. Ke své činnosti však používá relační databázi, takže je nutná konfigurace a spuštění MySQL serveru. O založení databáze a tabulek se stará již samotná aplikace.

K sestavení aplikace je potřeba Java 11 (JDK), Maven a Docker. V rootu aplikace je skript run_docker.sh, který musí být spuštěn v příkazové řádce (Bash). Před samotným spuštěním musí být změněna přístupová práva k tomuto souboru, aby byl skript spustitelný. Změna práv a spuštění skriptu je zobrazeno na ukázce příkazové řádky B.1.

Zdrojový kód B.1: Spuštění aplikace v Dockeru

```
chmod + x run_docker . sh
./ run_docker . sh
```

Pokud dojde ke správnému spuštění skriptu, tak se v Dockeru vytvoří nový kontejner mysqldb, do kterého je jako první nasazen a následně spuštěn image s MySQL serverem. Následně je sestavena aplikace, vytvořen image, nasazen do nového kontejneru app a automaticky spuštěn [6].

B.2 Spuštění aplikace v Dockeru

Po spuštění aplikace může uživatel přistoupit na URL adresu `http://localhost:8080` a pokračovat přihlášením do aplikace. V případě neexistence uživatelského účtu se musí nejdříve registrovat. Na registrační formulář se lze dostat z úvodní obrazovky stejně jako na formulář pro zapomenuté heslo. Po úspěšném přihlášení lze aplikaci plně využívat. Je připraveno i úložiště Data Lakehouse. Byly založeny dvě Delta tabulky (pro záznamy typu DASTA a RESQ) a do nich nahrány příslušné medicínské záznamy. Při automatickém sestavení a spuštění aplikace Dockerem je připravené úložiště nahráno do kontejneru s aplikací. V průběhu přípravy testovacího úložiště bylo potřeba aplikovat regulární výrazy, takže základní sada výrazů se vytváří při inicializaci aplikace a je aplikována na obě tabulky. Pro ověření funkčnosti je připravena i sada XQuery výrazů a nechybí ani sada SQL dotazů, která je dostupná pod sekcí Seznam SQL dotazů [6].

Aplikace využívá vizualizaci poskytovaného API pomocí modulu SpringDoc OpenAPI UI. Po přístupu na URL adresu `http://localhost:8080/swagger-ui/index.html` jsou viditelné všechny endpointy a používané transportní objekty. Vizualizace je dostupná jen po autentizace a odkaz je dostupný i skrze administraci [6].

B.3 Spuštění aplikace v lokálním prostředí

Po dobu vývoje a testování byla aplikace spouštěna ve vývojovém prostředí a MySQL databáze běžela na lokálním stroji, narozdíl od předchozí práce. Nejdříve je nutné změnit cestu k databázi, která je definována v souboru `application.properties` – cesta je zde připravena, takže ji stačí odkomentovat. Je potřeba mít nainstalovanou databázi MySQL, ve které je následně potřeba vytvořit databázi s názvem, který se nachází v URI pro připojení v `application.properties`, například příkazem `create database 'data-lakehouse'`. Po jejím vytvoření je možné aplikaci normálně používat.

Uživatelská dokumentace

C

V této dokumentaci jsou popisována pouze rozšíření aplikace Data Lakehouse. Pro popis zbylých funkcí aplikace je opět potřeba přečíst původní práci.

C.1 Změna názvů sloupců

Názvy sloupců je možné měnit na obrazovce "Úprava tabulky". Tam je možné se dostat z menu stitknutím tlačítka "Seznam tabulek"a následně výběrem tabulky v seznamu tlačítkem "Upravit". Nad komponentami pro vizualizaci schématu tabulky se nachází tlačítko "Správa názvu sloupců"a po kluknutí na něj se zobrazí modální okno, ve kterém je možné názvy sloupců upravit. Tyto názvy se poté zobrazí všude, kde jsou názvy sloupců viditelné. Jedná se o čistě estetický efekt, názvy sloupců tabulky v databázi zůstávají nezměněny.

C.2 Logistická regrese

Logistickou regresi lze provést na obrazovce "Datová analýza". Je nutné si nejdříve vybrat libovolnou tabulku, pro niž se následně zobrazí statistiky a výběr sloupců pro logistickou regresi. Je nutné si vybrat sloupce pro analýzu, poté je nutné vybrat kliknutím na vybraný sloupec třídu, pro kterou bude analýza provedena. Poté je možné stisknout tlačíko "Analyzovat". Následně se uživateli zobrazí graf s vypočítanými hodnotami a hodnoty budou také vypsané nad tabulkou. Není možné vybírat celé datové struktury. Pokud ji uživatel k analýze vybere, je programem upozorněn. Pokud uživatel vybere k analýze sloupec s typem `pole`, je zobrazeno modalové okno, ve kterém musí uživatel vybrat typ operace, kterým se pole bude agregovat, jinak analýza nebude fungovat.

C.3 Odvozování tabulek

Na stejné stránce "Datová analýza"je možné provést odvození tabulky tlačítkem "Odvození tabulky", na ní je nutné vybrat alespoň jednu tabulku a sloupce pro odvození. V případě, že uživatel vybere tabulky 2, je potřeba vybrat sloupec pro propojení a typ JOINu.

Libovolně je také možné vybrat podmínky, podle kterých budou jednotlivé sloupce filtrovány. Filtry a celý postup pro odvozování tabulek je také popsán v kapitole 7.3.

C.4 Statistiky

Statistiky se také zobrazují na obrazovce "Datová analýza". Statistiky se aktualizují po každém vložení dat do tabulky, statistiky odvozených tabulek se aktualizují po každém vložení dat do podkladových tabulek. Existuje také možnost vynutit statistiky okamžitě přepočítat, a to tlačítkem "Přepočítat statistiky". Je možné kliknout na jednotlivé řádky statistik, poté se zobrazí statistiky vybraného řádku (což odpovídá sloupci tabulky). Tabulku je možné řadit kliknutím na název sloupce v hlavičce tabulky. Pokud je vybraný sloupec pro řazení, je také možné zobrazit tabulku se statistikami jednotlivých sloupců. Toho je možné dosáhnout kliknutím na tlačítko "Zobrazit tabulku"pod tlačítkem "Přepočítat statistiky".

Bibliografie

1. BEN LORICA Michael Armbrust, et al. *What Is a Lakehouse?* [online]. Databricks Inc., 2020. [cit. 2024-01-06]. Dostupné z: https://www.databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html.
2. MILOSLAVSKAYA, Natalia; TOLSTOY, Alexander. Big Data, Fast Data and Data Lake Concepts. *Procedia Computer Science*. 2016.
3. *What Is Apache Parquet?* [online]. Dremio.com. [cit. 2024-01-15]. Dostupné z: https://www.dremio.com/resources/guides/intro-apache-parquet/.
4. *What Is a Data Warehouse?* [online]. Oracle.com. [cit. 2024-01-16]. Dostupné z: https://www.oracle.com/cz/database/what-is-a-data-warehouse/.
5. ARMBRUST, Michael; GHODSI, Ali; XIN, Reynold; ZAHARIA, Matei. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In: *Proceedings of CIDR*. 2021, sv. 8, s. 28.
6. MOUČKA, Lukáš. *Koncept Data Lakehouse pro zpracovaní medicínských dat*. 2023. Dostupné také z: http://hdl.handle.net/11025/53730.
7. HABÁŇ, Jaromír. *Technologie datových skladů* [online]. Cvis.cz, 2004. [cit. 2024-01-16]. Dostupné z: http://cvis.cz/hlavni.php?stranka=novinky/clanek.php%5C&id=60.
8. SMALLCOMBE, Mark. *ETL vs ELT: 5 Critical Differences* [online]. Integrate.io, 2023. [cit. 2024-01-07]. Dostupné z: https://www.integrate.io/blog/etl-vs-elt/.
9. *Build Lakehouses with Delta Lake* [online]. Delta Lake. [cit. 2024-04-03]. Dostupné z: https://delta.io/.
10. *Data Lakehouse* [online]. Databricks Inc. [cit. 2024-03-29]. Dostupné z: https://www.databricks.com/glossary/data-lakehouse.
11. *Medallion Architecture* [online]. Databricks Inc. [cit. 2024-04-22]. Dostupné z: https://www.databricks.com/glossary/medallion-architecture.

12. DATABRICKS. *What Is Apache Spark?* [online]. Databricks Inc. [cit. 2024-04-03]. Dostupné z: https://www.databricks.com/glossary/what-is-apache-spark.
13. TEAM, The Thymeleaf. *Thymeleaf*. The Thymeleaf Team, 2023. Dostupné také z: https://www.thymeleaf.org/.
14. *What is jQuery?* [online]. OpenJS Foundation. [cit. 2024-04-03]. Dostupné z: https://jquery.com/.
15. *Vývoj DASTA* [online]. Česká společnost zdravotnické informatiky avědeckých informací. [cit. 2024-03-29]. Dostupné z: https://dastacr.cz/.
16. *About DICOM: Overview* [online]. The Medical Imaging Technology Association. [cit. 2024-03-29]. Dostupné z: https://www.dicomstandard.org/about-home.
17. AWATI, Rahul; LAWTON, George. *XML Schema Definition (XSD)* [online]. TechTarget, 2022. [cit. 2024-01-13]. Dostupné z: https://www.techtarget.com/whatis/definition/XSD-XML-Schema-Definition.
18. *What is logistic regression?* [online]. IBM. [cit. 2024-01-16]. Dostupné z: https://www.ibm.com/topics/logistic-regression.
19. *Getting Started - Create a Chart* [online]. Chart.js, 2024. [cit. 2024-04-22]. Dostupné z: https://www.chartjs.org/docs/latest/getting-started/.
20. BOSTOCK, Mike. *What is D3?* [online]. Observable Inc. [cit. 2024-04-22]. Dostupné z: https://d3js.org/what-is-d3.
21. ZOKAEVA, Bella. *A Guide to the Best Java Machine Learning Libraries* [online]. OrbitSoft, 2024. [cit. 2024-01-11]. Dostupné z: https://orbitsoft.com/blog/best-java-machine-learning-libraries/.
22. RAFF, Edward. *JSAT: Java Statistical Analysis Tool, a Library for Machine Learning* [online]. JLMR.org, 2017. [cit. 2024-01-11]. Dostupné z: https://www.jmlr.org/papers/v18/16-131.html.
23. ALICE. *Machine Learning with Java libraries* [online]. Data Side Of Life, 2018. [cit. 2024-01-11]. Dostupné z: http://datasideoflife.com/?p=272.
24. *Spark Dataset Tutorial – Introduction to Apache Spark Dataset* [online]. DataFlair Web Services Pvt Ltd. [cit. 2024-04-03]. Dostupné z: https://data-flair.training/blogs/apache-spark-dataset-tutorial/.
25. *Class Attribute* [online]. University of Waikato. [cit. 2024-04-22]. Dostupné z: https://weka.sourceforge.io/doc.dev/weka/core/Attribute.html.

26. VCELAK, Petr. *Medical Research and Education* [online]. Medical Information Systems Research Group, 2024. [cit. 2024-03-21]. Dostupné z: `https://mre.zcu.cz/`.

27. *What Is a Data Dictionary?* [online]. The Regents of the University of California. [cit. 2024-04-19]. Dostupné z: `https://library.ucmerced.edu/data-dictionaries`.

Seznam obrázků

Seznam výpisů