Univerzita Jana Evangelisty Purkyně
v Ústí nad Labem
Přírodovědecká fakulta

Využití open-source a komerčních nástrojů pro vizualizaci a analýzu dat na datové platformě Portabo

BAKALÁŘSKÁ PRÁCE

Vypracoval: Ladislav Tahal

Vedoucí práce: Ing. Roman Vaibar, Ph.D., MBA

Studijní program: Aplikovaná informatika

Studijní obor:

Ústí nad Labem 2025

Namísto žlutých stránek vložte digitálně podepsané zadání kvalifikační práce poskytnuté vedoucím katedry.
Zadání musí zaujímat právě dvě strany.

Zadání je nutno vložit jako PDF pomocí některého nástroje, který umožňuje editaci dokumentů (se zachováním elektronického podpisu).

V Linuxe lze například použít příkaz `pdftk`.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a použil jen pramenů, které cituji a uvádím v přiloženém seznamu literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., ve znění zákona č. 81/2005 Sb., autorský zákon, zejména se skutečností, že Univerzita Jana Evangelisty Purkyně v Ústí nad Labem má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Jana Evangelisty Purkyně v Ústí nad Labem oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladu, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

V Ústí nad Labem dne 19. listopadu 2025 Podpis: . . . . . .

Děkuji vedoucímu bakalářské práce Ing. Romanu Vaibarovi, Ph.D., MBA za jeho odborné vedení a praktické podněty, které výrazně přispěly k analýze dat, hledání efektivních řešení a k celkovému úspěšnému dokončení bakalářské práce.

Využití open-source a komerčních nástrojů pro vizualizaci a analýzu dat na datové platformě Portabo

Abstrakt:

Bakalářská práce se zabývá srovnáním open-source a komerčních nástrojů pro vizualizaci a analýzu dat, datové sklady a OLAP technologie s cílem zhodnotit jejich vhodnost pro využití v datové platformě Portabo. V praktické části byla realizována implementace několika variant datových řešení, zahrnujících kombinace databázových systémů MSSQL, MariaDB, ClickHouse a PostgreSQL s nástroji Power BI, Superset a Cube.js. Bylo provedeno testování výkonu, analýza technických požadavků, nároků na uživatelské dovednosti a srovnání ekonomických aspektů provozu. Výsledkem práce je komplexní přehled výhod a nevýhod jednotlivých přístupů, měření jejich efektivity při zpracování velkých objemů dat a doporučení optimálního řešení pro organizace usilující o efektivní datovou analytiku bez nutnosti vysoké IT expertizy.

Klíčová slova: Business Intelligence, datové sklady, olap, vizualizace dat, Portabo

Utilization of Open-Source and Commercial Tools for Data Visualization and Analysis on the Portabo Data Platform

Abstract:

This bachelor's thesis focuses on the comparison of open-source and commercial tools for data visualization and analysis, data warehouses, and OLAP technologies, with the aim of evaluating their applicability within the Portabo data platform. In the practical part, several data architecture variants were implemented, combining database systems such as MSSQL, MariaDB, ClickHouse, and PostgreSQL with visualization tools including Power BI, Superset, and Cube.js. The analysis covers performance testing, technical requirements, user skill demands, and economic aspects of operation. The outcome of the thesis is a comprehensive overview of the advantages and limitations of both open-source and commercial solutions, performance evaluation on large data volumes, and recommendations for organizations seeking efficient data analytics without requiring extensive IT expertise.

Keywords: Business Intelligence, data warehouses, OLAP, data visualization, Portabo

Obsah

Úvod

V současné době organizace generují a shromažďují obrovské množství dat, která se stávají klíčovým zdrojem pro strategické i operativní rozhodování. Společnosti napříč odvětvími se proto zaměřují na to, jak tato data efektivně zpracovávat, ukládat, analyzovat a vizualizovat tak, aby přinášela skutečnou informační hodnotu i uživatelům bez hlubokého technického zázemí. S rozvojem datových technologií vzniká široké spektrum nástrojů, které umožňují tvorbu reportů, analytických modelů a vizualizací nad velkými objemy dat. Tyto nástroje mohou být jak komerční, nabízející komplexní podporu a integrované služby, tak open-source, které poskytují flexibilitu, otevřenost a nižší náklady na implementaci.

Zároveň však s tímto rozvojem vyvstává otázka, jaké řešení je pro konkrétní organizaci nejvhodnější – z pohledu technického, uživatelského i ekonomického. Volba správného nástroje či architektury datové platformy zásadně ovlivňuje nejen efektivitu zpracování dat, ale také dostupnost a interpretaci výsledků pro koncové uživatele.

Tato bakalářská práce se zabývá využitím open-source a komerčních nástrojů pro vizualizaci a analýzu dat na datové platformě Portabo. Cílem práce je provést srovnávací analýzu vybraných řešení z hlediska technických parametrů, uživatelských požadavků a ekonomických aspektů jejich provozu. Praktická část je zaměřena na implementaci několika variant datové architektury — zahrnujících kombinace databázových systémů MSSQL, MariaDB, ClickHouse a PostgreSQL s nástroji Power BI, Superset a Cube.js. Součástí analýzy je také měření výkonu při práci s rozsáhlým datovým souborem, testování zátěže a vyhodnocení celkových nákladů na provoz (TCO).

První kapitola práce shrnuje současný stav problematiky a základní pojmy z oblasti datových skladů, OLAP a vizualizačních nástrojů. Následující teoretická část rozebírá principy a architekturu zvolených technologií. Praktická část popisuje návrh metodiky, implementaci jednotlivých řešení a provedení testování. Závěrečná kapitola obsahuje shrnutí zjištěných výsledků, jejich interpretaci a doporučení vhodného řešení.

1. Přehled současného stavu problematiky

1.1. Rešerše v oblasti datových skladů, OLAP technologií a nástrojů pro vizualizaci dat

Tato kapitola přináší přehled současného stavu v oblasti zpracování, ukládání a vizualizace dat. Cílem je zmapovat, jak jsou dnes řešeny datové platformy určené pro analytické účely, jaké nástroje se používají v praxi a jaké trendy určují směr vývoje v oblasti datové analytiky.

Datové sklady a vývoj analytických databází

Datové sklady a technologie OLAP patří k dlouhodobě stabilním základům datové analytiky. Již několik desetiletí tvoří klíčovou infrastrukturu pro zpracování a konsolidaci dat z různých zdrojů, přičemž jejich architektura se neustále vyvíjí s ohledem na rostoucí objem dat a potřebu vyšší výpočetní efektivity.

Původně byly analytické systémy budovány nad relačními databázemi. Tyto systémy dominovaly zejména podnikovému prostředí a poskytovaly základní podporu pro tvorbu datových skladů. Postupně se však začala prosazovat specializovaná řešení určená přímo pro analytické zpracování velkých objemů dat.

Moderní analytické databáze, jako jsou *ClickHouse*, *Snowflake*, *Amazon Redshift* či *Google BigQuery*, využívají kolumnární uložení dat, paralelní zpracování a škálovatelnou cloudovou architekturu [1, 2]. Výhodou těchto systémů je možnost provádět rozsáhlé analytické dotazy v reálném čase a minimalizovat potřebu předběžných agregací. Vedle komerčních řešení se rozvíjí i open-source alternativy, které kombinují vysoký výkon s nízkými náklady na provoz – například *ClickHouse* či *Apache Druid*.

OLAP technologie a analytické přístupy

OLAP technologie zůstávají jedním z hlavních způsobů, jak efektivně analyzovat a agregovat data pro potřeby rozhodování. Tradiční přístupy založené na předpočítaných datových kostkách jsou dnes doplňovány flexibilnějšími modely, které umožňují ad-hoc analýzy nad velkými datovými sadami.

V současnosti lze pozorovat trend integrace OLAP funkcionality přímo do datových skladů. Například platformy jako *Snowflake*, *ClickHouse* nebo *PostgreSQL* s rozšířením *TimescaleDB* umožňují analytické dotazování bez nutnosti samostatné OLAP vrstvy [1]. Výsledkem je zjednodušená architektura a nižší náklady na údržbu.

OLAP se zároveň posouvá směrem k real-time zpracování, kde je možné kombinovat streamovaná a historická data. Tento přístup umožňuje okamžitou reakci na události, což je zásadní například v oblasti IoT, průmyslového monitoringu nebo dopravní analýzy.

Nástroje pro vizualizaci a business intelligence

Vizualizační a BI nástroje představují prezentační vrstvu datové analytiky. Umožňují nejen tvorbu reportů a dashboardů, ale i interaktivní exploraci dat bez nutnosti pokročilých programátorských znalostí. Nároky na uživatele se liší v závislosti na architektuře řešení. V případě přímého napojení na databázi může být vyžadována znalost jazyka SQL. Pokud je však využita sémantická vrstva, uživatel pracuje již s předpřipravenými datovými objekty, které pouze vizualizuje. Pro pokročilejší úpravy či specifické kalkulace lze následně využít specializované jazyky daného nástroje, jako jsou například DAX pro výpočty nebo Power Query pro transformaci dat v prostředí Power BI Na trhu existuje široké spektrum nástrojů, od komerčních platforem po open-source řešení. Mezi nejpoužívanější komerční systémy patří *Microsoft Power BI*, *Tableau* a *Qlik Sense*. Tyto produkty nabízejí komplexní ekosystém pro tvorbu vizualizací, datových modelů a propojení s různými datovými zdroji.

Z open-source nástrojů se výrazně prosadily *Apache Superset*, *Grafana* a *Metabase*, které poskytují flexibilní možnosti přizpůsobení, automatizace a integrace.Zajímavým směrem vývoje je i koncept tzv. *sémantické vrstvy* (*semantic layer*). Ta slouží jako překladový můstek mezi technickou strukturou databáze a byznysovým pohledem uživatele. Reprezentantem tohoto přístupu je například framework *Cube.js*, který umožňuje firmám zpřístupnit datový sklad i uživatelům bez hluboké znalosti SQL. [3, 4, 5].

1.2. Přehled současných open-source a komerčních řešení

V současné době existuje řada komplexních ekosystémů určených pro správu, zpracování a analýzu dat. Tyto ekosystémy zpravidla zahrnují nástroje pro integraci dat (ETL/ELT), datové sklady, OLAP

vrstvy a vizualizační rozhraní. Cílem této části je poskytnout rešeršní přehled nejrozšířenějších řešení, která se využívají při budování datových platforem.

Před samotným přehledem je důležité vymezit klíčové pojmy, které budou v práci dále používány. Jedná se především o dělení na **komerční** a **open-source** software a dále na **on-premise** a **cloudová** řešení.

Klasifikace řešení

- **Komerční software** je software, jehož distribuce a používání je podmíněno zakoupením licence. Jeho **zdrojový kód je uzavřený (proprietární)**, což znamená, že uživatelé nemají kód k dispozici, nemohou jej prohlížet, modifikovat, ani provádět zpětné inženýrství. Důsledkem je **nemožnost modifikovat** chování systému a potenciální **závislost na dodavateli** v otázkách úprav a oprav. Komerční řešení bývají vyvíjena a distribuována za účelem zisku. Výhodou bývá profesionální podpora, garantovaná kvalita a ucelený ekosystém nástrojů.
- **Open-source software** poskytuje uživatelům přístup k veřejně dostupnému zdrojovému kódu a uděluje jim práva k jeho používání, studování, modifikaci a šíření. Klíčovým prvkem je zde **open-source licence**, která přesně definuje, jakým způsobem lze s dílem a jeho modifikacemi nakládat. **Detailní pochopení licenčních podmínek je pro praktické nasazení zásadní** (pro přehled a podmínky licencí je nutné nahlédnout do oficiální dokumentace konkrétní licence).
  - *Permisivní licence* (např. **MIT**, **Apache 2.0**) jsou maximálně benevolentní, kladou minimální omezení a obvykle umožňují upravený kód uzavřít a následně komerčně prodávat jako proprietární produkt.
  - *Copyleftové licence* (např. **GPL**) jsou přísnější a vyžadují, aby jakékoliv odvozené dílo nebo úprava byly šířeny pod stejnou svobodnou licencí. Tím je zajištěno, že zdrojový kód zůstane trvale veřejně přístupný i v budoucích verzích.

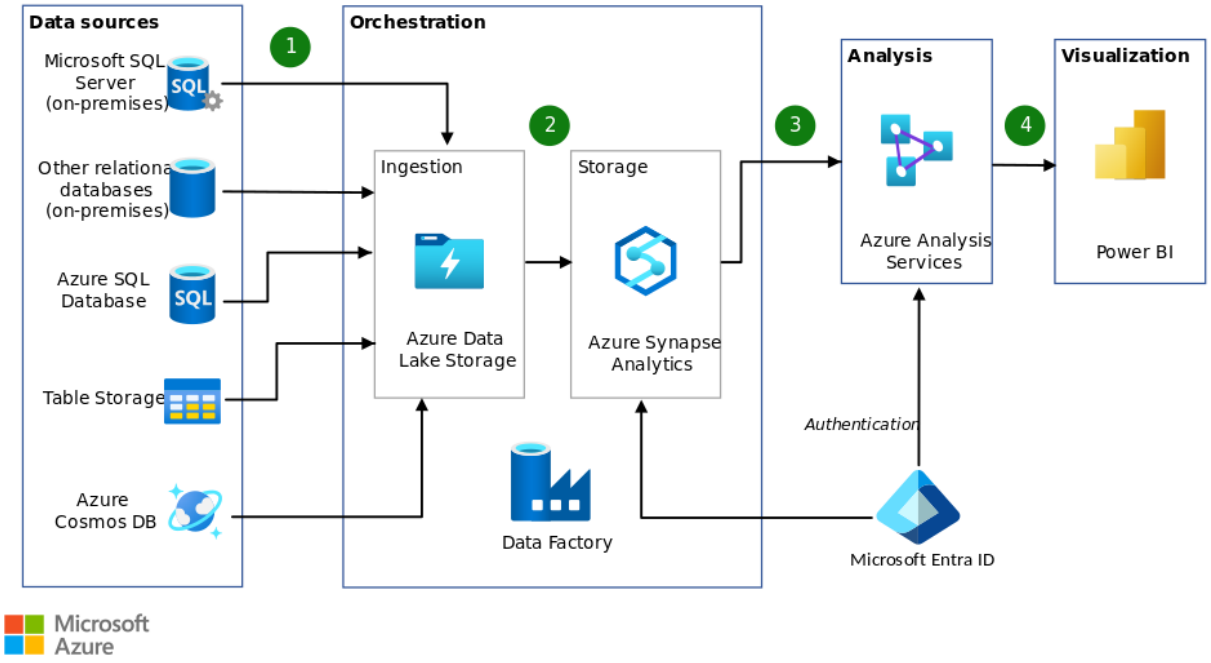
  Mezi obecné výhody open-source softwaru patří nezávislost na dodavateli, nižší počáteční náklady a možnost bezpečnostního auditu komunitou. Nevýhodou mohou být vyšší nároky na technické znalosti při implementaci a nutnost řešit podporu buď komunitně, nebo placenou službou třetí strany.
- **On-premise řešení** je provozováno na vlastní infrastruktuře organizace (vlastních serverech). To poskytuje plnou kontrolu nad daty a systémem, ale zároveň vyžaduje vyšší počáteční investice do hardwaru a personálu pro jeho správu.
- **Cloudové řešení** je poskytováno jako služba (SaaS, PaaS, IaaS) třetí stranou (např. Microsoft Azure, AWS, Google Cloud). Výhodou je škálovatelnost, platba za skutečné využití (pay-as-you-go) a menší nároky na vlastní IT oddělení. Nevýhodou může být menší kontrola nad infrastrukturou a potenciální závislost na poskytovateli.

Tato klasifikace je zásadní pro pozdější srovnání celkových nákladů na vlastnictví (TCO), kde se projeví rozdíly v licenčních poplatcích, nákladech na infrastrukturu a správu.

Microsoft Data Platform (komerční, on-premise/cloud)

Ekosystém společnosti Microsoft představuje jedno z nejkomplexnějších řešení pro podnikové zpracování dat, které pokrývá celý životní cyklus od datových toků až po vizualizaci. Jeho unikátnost spočívá v možnosti nasazení jak v tradičním on-premise prostředí, tak v cloudu, případně v hybridním režimu. Komponenty platformy lze rozdělit následovně:

- **Tradiční on-premise infrastruktura:** Základem je *Microsoft SQL Server*. Pro integraci dat (ETL) je využíván modul *SQL Server Integration Services* (SSIS) a pro analytické modelování (OLAP) slouží *SQL Server Analysis Services* (SSAS). Toto složení tvoří osvědčený standard pro lokální datové sklady [6].
- **Cloudová platforma (Microsoft Azure):** S nástupem cloudu se architektura posouvá do prostředí Azure (viz Obrázek 1.1). Roli datového skladu zde přebírá *Azure Synapse Analytics*, ETL procesy zajišťuje *Azure Data Factory* a celková orchestrace se sjednocuje v rámci platformy *Microsoft Fabric*.



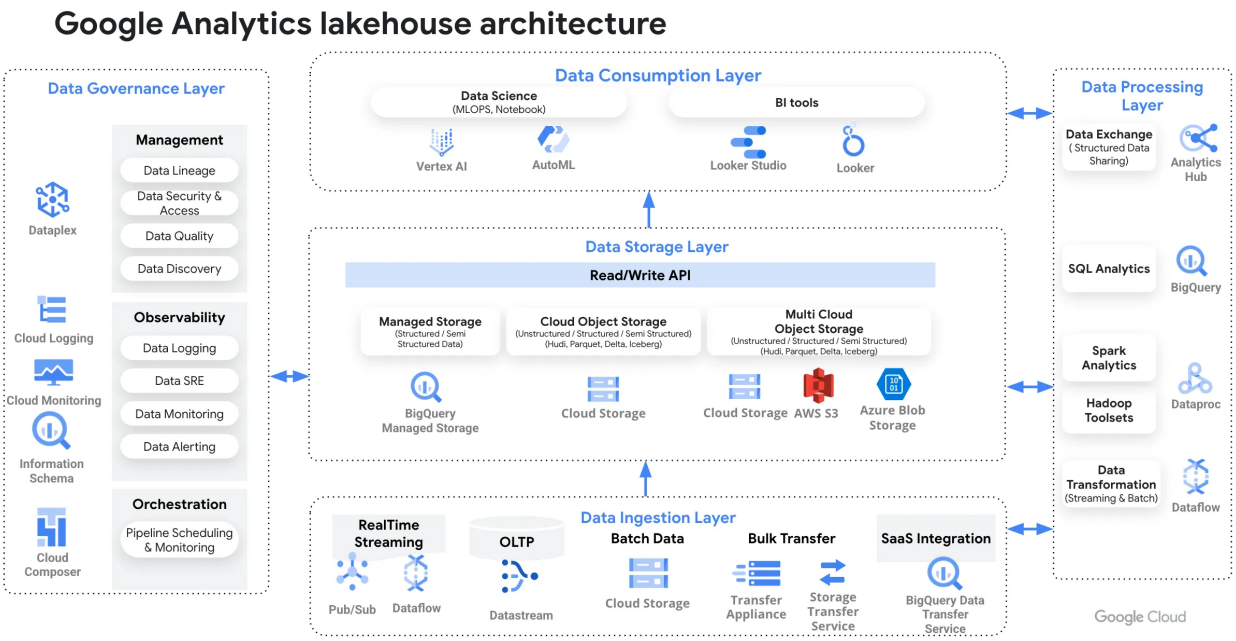
Obrázek 1.1.: Architektura moderního datového skladu v prostředí Microsoft Azure - převzato z [7]

- **Vizuální a sémantická vrstva:** Sjednocující vrstvou pro on-premise i cloud je *Power BI*. Tento nástroj slouží pro tvorbu reportů a dashboardů a díky své sémantické vrstvě dokáže zastoupit i některé funkce SSAS [8].
- **Výhody a nevýhody:** Hlavní výhodou je hluboká integrace všech komponent. Nevýhodou, zejména u cloudových služeb, může být silný „vendor lock-in“ a závislost na licenční politice poskytovatele.

Google Cloud Data Analytics (komerční, cloud)

Společnost Google přistupuje k analytice jako k plně cloudové službě, navržené od počátku pro distribuované výpočty. Klíčové komponenty ekosystému jsou:

- **Datový sklad:** Jádrem platformy je *Google BigQuery*, serverless datový sklad s kolumnárním uložením, kde se platí pouze za uložená data a zpracované dotazy [9].



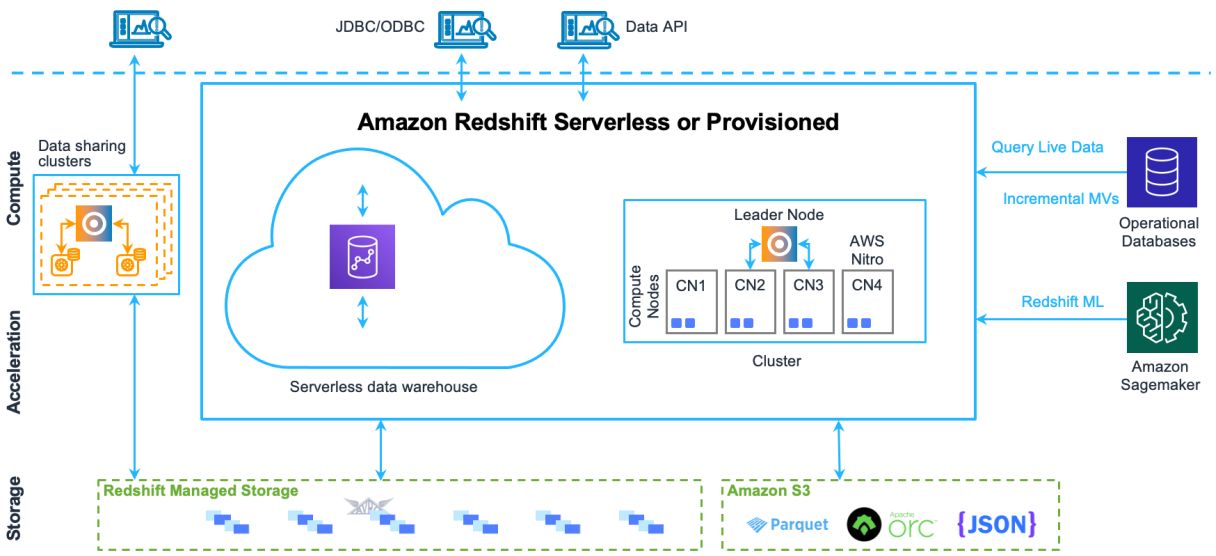
Obrázek 1.2.: Architektura analytické platformy Google Cloud - převzato z [10]

- **Integrace a příprava dat:** Pro tyto účely slouží služby jako *Dataflow* (založené na Apache Beam) nebo *Dataprep*.
- **Prezentační vrstva:** Zastupuje ji *Looker Studio* pro rychlou vizualizaci, případně platforma *Looker*, která nabízí pokročilou sémantickou vrstvu a governance dat.
- **Výhody a omezení:** Hlavní výhodou je nulová starost o hardware, vysoká škálovatelnost a integrace s AI nástroji (*Vertex AI*). Zásadním omezením je absence on-premise řešení, což znemožňuje využití organizacím, které vyžadují uložení dat na vlastním hardwaru.

Amazon Web Services (komerční, cloud)

Cloudová platforma *Amazon Web Services* (AWS) od společnosti Amazon nabízí rozsáhlý a modulární ekosystém pro datové inženýrství a analytiku.

- **Datový sklad:** Klíčovou roli zde zastává *Amazon Redshift* – cloudový datový sklad postavený na kolumnární architektuře [11].
- **Zpracování a vizualizace:** Služba *AWS Glue* umožňuje automatizované ETL procesy, zatímco *Amazon QuickSight* poskytuje prostředí pro interaktivní vizualizace a reporting.



Obrázek 1.3.: Cloudové řešení Amazon Web Services - převzato z [12]

- **Integrace s open-source:** AWS podporuje integraci s open-source systémy, například *Apache Spark* (prostřednictvím služby *EMR*) nebo *Presto*.
- **Výhody a slabiny:** Výhodou AWS je modularita, která umožňuje komponenty kombinovat podle potřeb. Podobně jako u Google se jedná o výhradně cloudové řešení. Slabinou může být větší komplexita nastavení a vyšší provozní náklady.

Ekosystém Apache Software Foundation

Nejvýznamnějším hráčem v oblasti otevřených datových technologií je nadace *Apache Software Foundation* (ASF). Její ekosystém de facto definoval standardy pro zpracování velkých dat (Big Data). Produkty pod hlavičkou Apache lze skládat do výkonné platformy pokrývající celý datový cyklus:

- **Ukládání a zpracování:** Historicky založeno na ekosystému *Hadoop*, dnes dominuje *Apache Spark* pro distribuované výpočty a *Apache Kafka* pro streamování dat.
- **OLAP a analytika:** Pro rychlé analytické dotazy slouží nástroje jako *Apache Druid* nebo *Apache Kylin*, který přináší schopnost multidimenzionální analýzy (OLAP kostky) na velká data, podobně jako SSAS v ekosystému Microsoft [**apachekylin2024**].
- **Vizualizace:** Prezentační vrstvu zajišťuje *Apache Superset*, moderní BI nástroj umožňující tvorbu dashboardů a granulární řízení přístupových práv.

Výhodou tohoto ekosystému je jeho modularita a fakt, že tvoří technologický základ i pro mnoho komerčních cloudových služeb (např. AWS EMR).

Platforma MariaDB (Community / Enterprise)

Příkladem open-source ekosystému, který se transformoval do podoby robustní podnikové platformy, je *MariaDB*. Ačkoliv původně vznikla jako fork databáze MySQL, ve své edici *MariaDB Enterprise Platform* dnes nabízí sadu integrovaných komponent pro transakční, analytické i hybridní zpracování dat a nově i pro umělou inteligenci.

Základem platformy je *MariaDB Enterprise Server*, který podporuje práci s relačními daty i formátem JSON. Síla ekosystému však spočívá v jeho specializovaných modulech a službách:

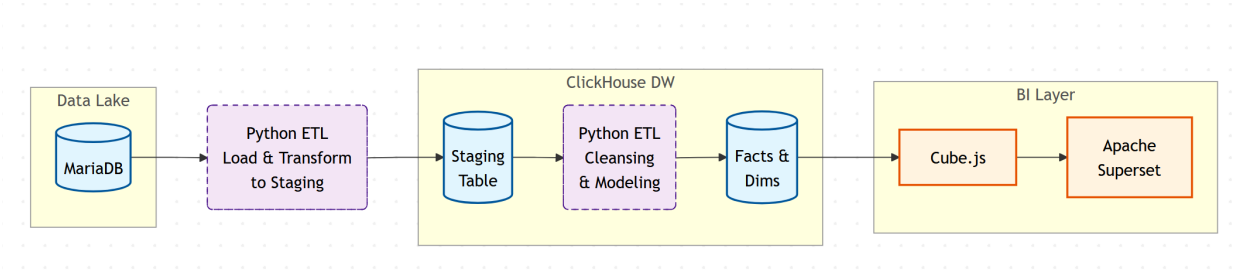
- **Analytika a Big Data:** Pro analytické úlohy slouží *MariaDB ColumnStore*, což je sloupcové úložiště umožňující rychlé agregace bez nutnosti vytvářet složité indexy. Novinkou je komponenta *MariaDB Exa*, in-memory MPP (Massively Parallel Processing) databáze navržená pro extrémně rychlé SQL dotazy nad velkými objemy dat. Propojení transakčního a analytického světa zajišťuje funkce *Query Accelerator*, která automaticky deleguje náročné dotazy do ColumnStore enginu.
- **Vysoká dostupnost a správa:** Klíčovým prvkem infrastruktury je *MariaDB MaxScale*. Jde o inteligentní proxy server, který zajišťuje load balancing, zabezpečení a automatické failover procesy. Pro orchestraci a monitorování celé flotily databází slouží *MariaDB Enterprise Manager*.
- **Integrace AI:** Platforma reaguje na nástup generativní umělé inteligence modulem *MariaDB AI RAG*, který umožňuje propojit firemní data s jazykovými modely (Retrieval-Augmented Generation). Rozhraní mezi AI asistenty a datovým ekosystémem pak zprostředkovává *MariaDB MCP Server*.

Tato architektura umožňuje organizacím začít s open-source řešením (Community verze) a v případě potřeby škálovat na plnohodnotnou enterprise platformu s podporou pro AI a real-time analytiku, aniž by musely migrovat na proprietární databázové systémy typu Oracle či MS SQL.

Moderní modulární a hybridní architektury

V současné praxi se stále častěji ustupuje od využívání jednoho monolitického „megabalíku“ (jako je kompletní Apache stack) směrem k flexibilním, modulárním architekturám. Tento trend umožňuje skládat platformu z několika specializovaných nástrojů od různých tvůrců, a to přesně podle potřeb daného projektu [2]. Tento přístup lze realizovat dvěma hlavními způsoby:

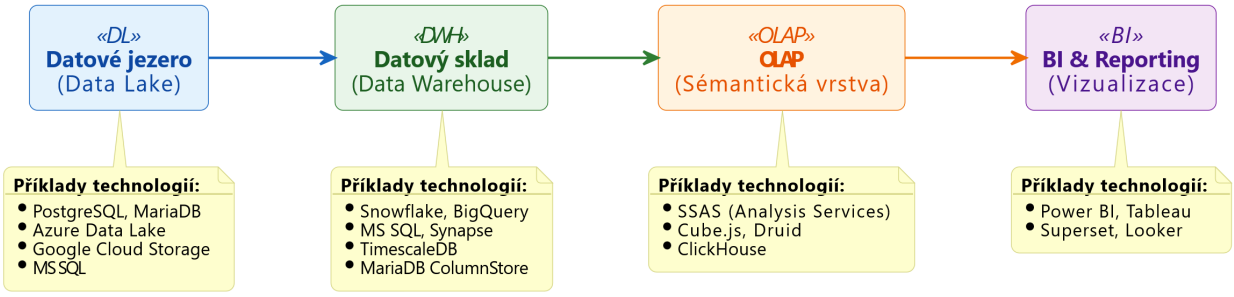
- **Čistě open-source stack:** Tento model kombinuje výhradně open-source komponenty. Typickým příkladem je použití relační databáze (např. *MariaDB*, *PostgreSQL*, či *ClickHouse*) jako datového skladu, nad kterým je postavena sémantická vrstva v podobě *Cube.js* nebo *dbt Semantic Layer*. Tyto nástroje definují metriky a business logiku, kterou následně konzumuje vizualizační nástroj jako *Apache Superset*, *Metabase* nebo *Grafana* [3, 13, 14, 15].



Obrázek 1.4.: Příklad možné plně open-source sestavy

- **Hybridní stack:** V praxi je velmi běžné kombinovat open-source a komerční technologie, aby se využily výhody obou světů. Častým scénářem je provozování výkonné a nákladově efektivní open-source databáze (např. *MariaDB*, *PostgreSQL*) jako datového skladu, nad nímž je nasazen uživatelsky přívětivý komerční vizualizační nástroj jako *Power BI* nebo *Tableau*. Tento přístup umožňuje optimalizovat náklady na back-end a zároveň poskytnout koncovým uživatelům komfort a pokročilé funkce komerčních platforem.

  Hlavní výhodou modulárních přístupů je vysoká míra škálovatelnosti, transparentnost a nezávislost na jednom dodavateli, což umožňuje reagovat na technologické změny a optimalizovat náklady. Společnou slabinou je však vyšší technická složitost. Integrace komponent z odlišných prostředí vyžaduje nejen znalost různých technologií, ale i pochopení jejich licenčních modelů a zajištění vzájemné kompatibility. Pro úspěšnou implementaci je proto klíčová standardizace rozhraní a pečlivé řízení verzí jednotlivých nástrojů.



Obrázek 1.5.: Příklad možného hybridního systému

1.3. Analýza trendů v implementaci

V posledních letech dochází v oblasti implementace datových platforem a systémů pro analýzu dat k významným změnám. Tyto změny jsou důsledkem rostoucích požadavků na rychlost zpracování, automatizaci a dostupnost analytických nástrojů i mimo oblast IT. Současný vývoj ukazuje posun od uzavřených, monolitických řešení směrem k otevřeným, modulárním a škálovatelným architekturám, které umožňují flexibilnější integraci technologií a snadnější rozšiřování podle potřeb organizace [16, 17].

Automatizace a ELT přístup

Jedním z hlavních trendů posledních let je přechod od tradičního přístupu *ETL* (Extract–Transform–Load) k modernějšímu konceptu *ELT* (Extract–Load–Transform). Transformace dat se tak přesouvá z úrovně samostatných procesů do samotného datového skladu, který disponuje dostatečným výkonem pro jejich zpracování. Tento přístup zjednodušuje datové toky, zvyšuje jejich transparentnost a umožňuje verzování datových modelů. Rozšířené je využití nástrojů jako *dbt*, *Apache Airflow* nebo *Fivetran*, které podporují automatizované a reprodukovatelné datové pipeline.

Datová governance a kvalita dat

S rostoucím objemem dat získává na významu jejich správa, kvalita a dohledatelnost. Moderní přístupy kladou důraz na principy *data governance*, které zahrnují řízení přístupových práv, dokumentaci datových toků (*data lineage*) a sledování kvality dat. K rozšířeným nástrojům patří například *Apache Atlas*, *Amundsen* či *DataHub*, které umožňují centralizovanou správu metadat. V komerční sféře pak obdobné funkce zajišťují systémy jako *Microsoft Purview* nebo *Collibra*. Implementace těchto principů je klíčová nejen z hlediska efektivity, ale i souladu s legislativními požadavky, například GDPR nebo ISO 27001.

Dotazování přirozeným jazykem a role LLM

V posledních letech lze pozorovat výrazný rozvoj velkých jazykových modelů (LLM, *Large Language Models*) a jejich integraci do analytických platforem. Tento trend, často označovaný jako *Natural Language Querying* (NLQ), slibuje revoluci ve způsobu, jakým uživatelé interagují s daty. Místo psaní složitých SQL dotazů nebo manuálního proklikávání dashboardů mohou uživatelé pokládat otázky v přirozeném jazyce, a to i hlasem.

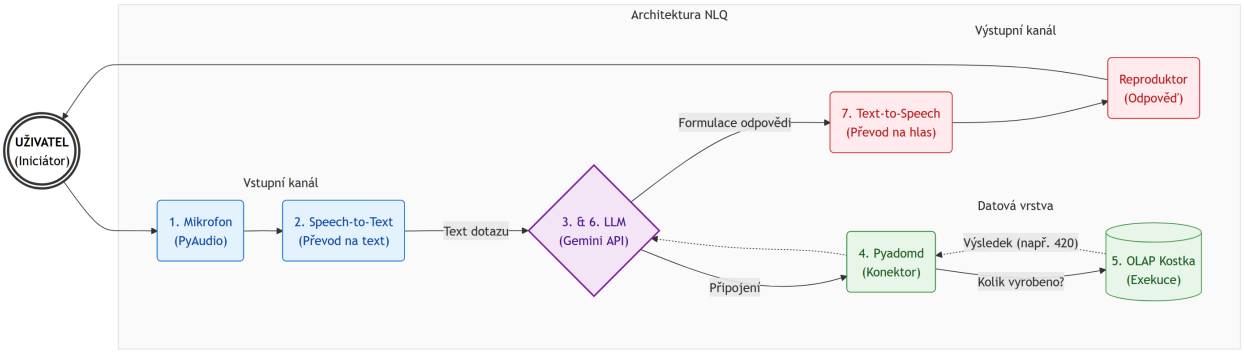
Architektura řešení

Systém pro dotazování přirozeným jazykem je komplexní architektura složená z několika komponent, které spolupracují na transformaci lidské řeči na strojově čitelný dotaz a zpět na srozumitelnou odpověď. Typický proces lze popsat v následujících krocích:

1. **Zpracování hlasového vstupu:** Uživatel položí dotaz hlasem, například: „Zjisti, kolik tun papíru bylo vyrobeno za poslední týden.“ Tento vstup je zachycen mikrofonem a zpracován pomocí knihovny jako **PyAudio**.
2. **Převod řeči na text (Speech-to-Text):** Zvukový záznam je odeslán do služby pro převod řeči na text, která jej transformuje do textové podoby.
3. **Interpretace dotazu pomocí LLM:** Získaný text je předán velkému jazykovému modelu, například přes **Gemini API**. Model analyzuje text, identifikuje záměr uživatele a klíčové entity (metriku „tuny papíru“, časové období „poslední týden“).

4. **Plánování a volání nástrojů:** Na základě interpretace LLM naplánuje, jaké kroky jsou potřeba k získání odpovědi. To typicky zahrnuje volání externích nástrojů – v tomto případě dotaz do databáze. Toto volání se neděje napřímo, ale prostřednictvím specializované mezivrstvy (viz níže).
5. **Exekuce dotazu:** Požadavek od LLM je přeložen na konkrétní SQL nebo DAX dotaz a vykonán v cílovém datovém skladu nebo OLAP kostce (např. pomocí knihovny **Pyadomd** pro komunikaci s MS SSAS).
6. **Zpracování výsledku a formulace odpovědi:** Výsledek dotazu (např. číslo 420) je vrácen LLM, který jej interpretuje a zformuluje odpověď v přirozeném jazyce: „Za poslední týden bylo vyrobeno 420 tun papíru."
7. **Převod textu na řeč (Text-to-Speech):** Odpověď může být převedena zpět do zvukové podoby a přehrána uživateli.

Celý tento proces elegantně skrývá technickou složitost a poskytuje uživateli iluzi plynulé konverzace s datovým systémem.



Obrázek 1.6.: Návrh NLQ architektury a teoretického využití v korporátním prostředí

Role MCP serverů a Gemini CLI

Klíčovou komponentou pro bezpečnou a spravovatelnou komunikaci mezi LLM a externími systémy (databáze, soubory, API) jsou tzv. **MCP (Multi-Context Prompt) servery**. Tyto servery fungují jako řízená brána neboli „toolbox" pro umělou inteligenci. Místo toho, aby měl LLM přímý a nekontrolovaný přístup k datovým zdrojům, MCP server mu poskytuje sadu jasně definovaných **nástrojů** (tools). Každý nástroj reprezentuje konkrétní operaci, například „spusť SQL dotaz v databázi ClickHouse" nebo „přečti soubor z Google Drive".

Pro interakci s těmito servery a pro definici toho, jaké nástroje a kontext má AI k dispozici, se používají nástroje příkazové řádky jako **Gemini CLI**. Tento nástroj umožňuje vývojáři „uzemnit"

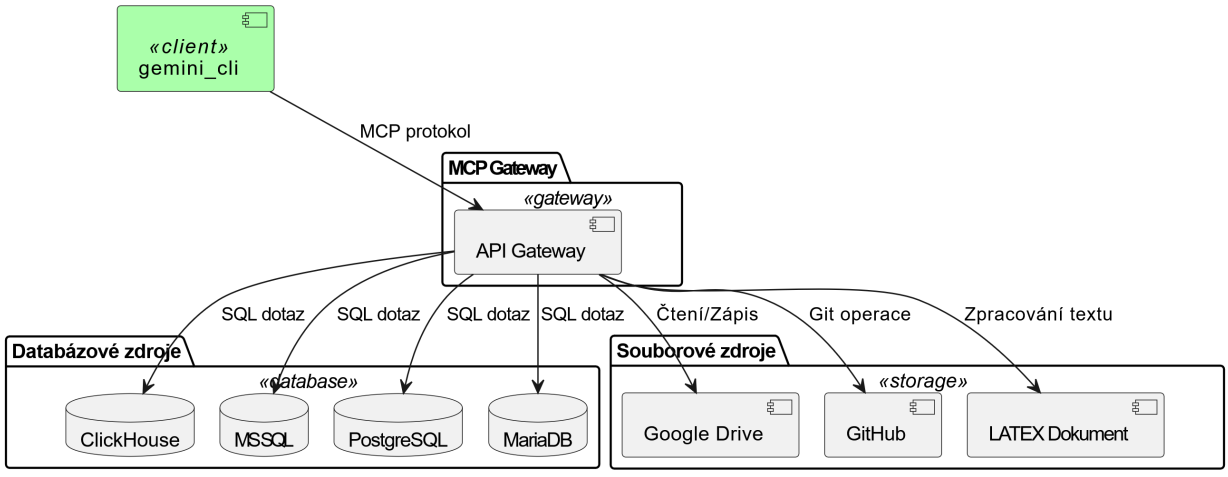
(ground) model tím, že mu poskytne veškeré potřebné informace a oprávnění pro řešení daného úkolu. LLM se pak nedotazuje přímo databáze, ale volá nástroje zpřístupněné přes MCP server, který se stará o samotnou exekuci, autentizaci a logování.

Kontext a praktické využití pro revizi práce

Samotný LLM bez dodatečných informací nedokáže odpovídat na specifické otázky týkající se konkrétního projektu. Aby mohl efektivně pracovat, musí mu být poskytnuta **paměť neboli kontext**. V mém případě jsem pro revizi a kontrolu konzistence této bakalářské práce využil právě **Gemini CLI**. Tímto nástrojem jsem AI poskytl přístup k veškerým relevantním zdrojům:

- **Zdrojové kódy práce** a související soubory uložené na **Google Drive** a v repozitáři na **GitHubu**.
- Přímé napojení na testovací **databáze** (ClickHouse, MS SQL, PostgreSQL, MariaDB) pro ověřování informací o jejich struktuře a obsahu.
- Samotný LaTeX soubor této práce pro kontrolu textu a struktury.

Díky tomuto rozsáhlému kontextu jsem mohl pokládat komplexní dotazy jako: „Jsem ve své práci konzistentní v pojmenování ETL skriptů?“ nebo „Odpovídá popis architektury v kapitole 3 implementaci v mých Docker souborech?“. Ačkoliv má Gemini kontextové okno o velikosti 1 milionu tokenů, pro komplexní analýzu je stále nutné pokládat dotazy strategicky. Například místo dotazu na kompletní obsah databáze (což by rychle vyčerpalo kontext) je efektivnější ptát se na její strukturu, indexaci nebo metadata, a tím nechat AI ověřit soulad mezi implementací a popisem v textu práce.



Obrázek 1.7.: Popis struktury pomocí UML komponent diagramu

2. Teoretická část

Tato kapitola se zaměřuje na teoretické vymezení klíčových pojmů a principů, které tvoří základ pro praktickou část práce. Cílem je popsat architekturu moderních datových platforem, jejich vrstvy a související technologie využívané pro ukládání, zpracování a analýzu dat. Pozornost je věnována především konceptům *datového jezera*, *datového skladu* a technologiím *OLAP*, které tvoří jádro současných analytických systémů. Součástí kapitoly je také přehled nástrojů Business Intelligence a teoretické vymezení *sémantické vrstvy*, jež propojuje datový sklad s prezentační částí platformy a umožňuje jednotnou interpretaci dat napříč organizací.

2.1. Úvod do problematiky datových skladů a OLAP

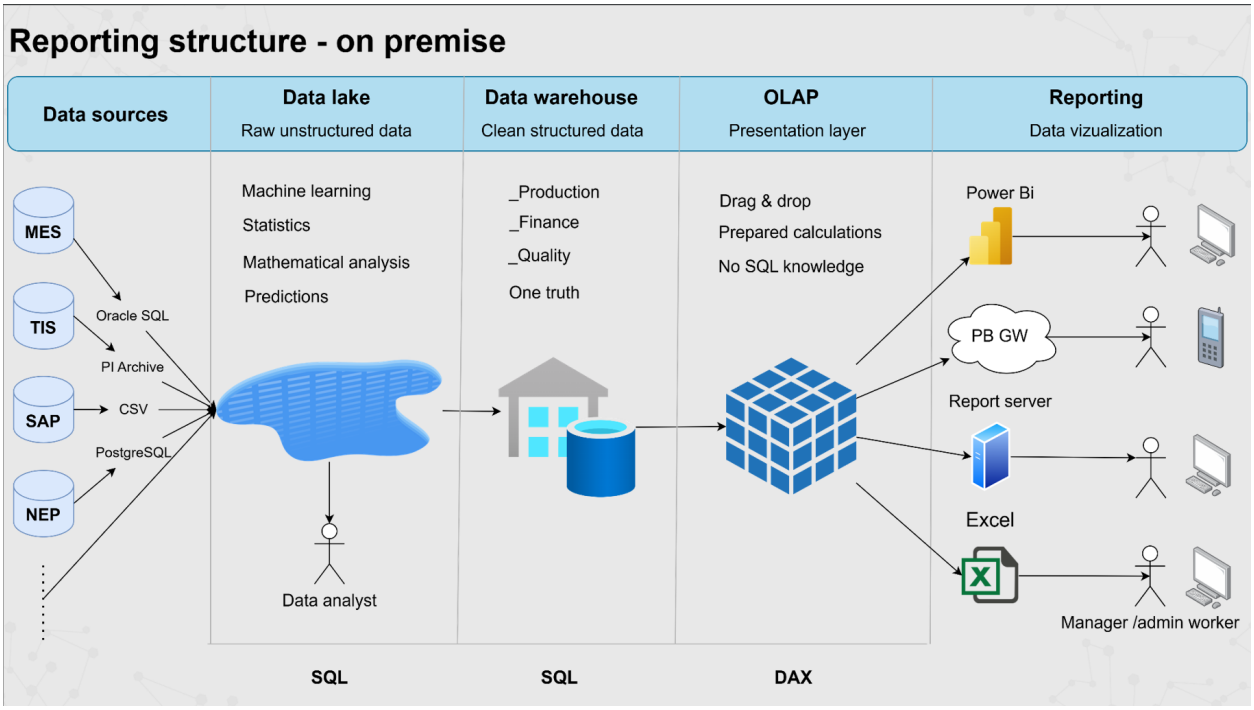
Návrh systému, který byl v této práci vytvořen, nepředstavuje univerzální ani definitivní řešení. Každý datový architekt by měl vždy zohlednit konkrétní potřeby a podmínky dané organizace. Například některé společnosti nemusí vyžadovat implementaci OLAP systému – mohou disponovat vlastním týmem databázových specialistů, kteří vytvářejí databázové pohledy, a datových analytiků, kteří z těchto pohledů následně generují reporty. Jiné organizace naopak tento systém nemohou použít vůbec, například v případě požadavku na zpracování dat v reálném čase, pro které jsou vhodnější jiné technologie a architektury.

Navržený systém proto představuje jedno z možných řešení, přizpůsobené konkrétní situaci a prostředí, ve kterém byl vyvíjen.

Celý systém je rozdělen do pěti vrstev, jak je znázorněno na obrázku 2.1.

Datové zdroje

Základní vstupní vrstvu systému tvoří datové zdroje, které obvykle pocházejí z různých podnikových systémů, jako jsou například MES systémy, CRM nebo ERP. Datový zdroj je zpravidla reprezentován databází, avšak v praxi se často setkáváme také s textovými soubory (například ve formátu CSV). Ačkoliv formát CSV nemusí na první pohled působit jako spolehlivý datový zdroj, v praxi se používá poměrně často. Důvodem bývá například licenční politika – přímý přístup do databáze může být zpoplatněn, zatímco export dat do CSV souboru bývá dostupný zdarma nebo bez omezení.



Obrázek 2.1.: Navržená struktura systému. Datové zdroje jsou ilustrační.

Datové jezero

Datové jezero (*Data Lake*) představuje logický koncept centralizovaného úložiště určeného pro uchovávání obrovského množství *surových dat* (*raw data*) v jejich nativním formátu (např. JSON, XML, binární soubory).

Ačkoliv je Datové jezero často asociováno s cloudovými službami (např. AWS S3, Azure Data Lake Storage), tento koncept lze efektivně implementovat i v lokálním (on-premise) prostředí.

V rámci řešeného projektu je Datové jezero implementováno na relační databázi (viz konfigurační soubor „docker-compose3.yml“). Přestože relační databáze vyžaduje definici schématu tabulky, je princip *schema-on-read* zachován. To je realizováno tak, že veškerá variabilní data jsou uložena v jediném sloupci (např. `payload`), který je definován jako řetězcový (textový) typ (`TEXT` nebo `VARCHAR`). Tím se fyzicky vytvoří pevné schéma pro tabulku, avšak **logický datový typ jednotlivých vnitřních prvků** dat (např. JSON) se určuje staticky / dynamicky **až v rámci transformačního (ETL) procesu** při jejich parsování a vkládání do Datového skladu. Tato volba zachovává maximální flexibilitu, ačkoliv pokročilejší implementace by mohla využít nativní datové typy pro nestrukturovaná data, jako je `JSONB` v PostgreSQL.

Klíčové role a cíle Datového jezera:

1. **Konsolidace a Vstupní Zóna (*Landing Zone*):** Primární funkcí je konsolidovat data z mnoha heterogenních zdrojů na jedinou platformu, sloužící jako **vstupní zóna** pro surová data před jejich dalším zpracováním.
2. **Oddělení Integrační Logiky:** Umožňuje **oddělit logiku připojení a sběru dat** od vlastního Datového skladu. Tím se zajišťuje, že náročné transformační procesy (ETL/ELT) v DWH

nejsou bezprostředně zatíženy problémy s konektivitou, dostupností zdrojů nebo dynamickou strukturou dat.

3. **Audit a Rodokmen Dat (*Data Lineage*):** Uchovávání dat v jejich **původním (surovém) formátu** umožňuje kdykoliv ověřit, z jakých zdrojových dat byla odvozena data v Datovém skladu. To je nezbytné pro **auditní účely** a pro **reprodukci analytických výsledků** při změnách transformačních pravidel.

Datový sklad

Datový sklad (*Data Warehouse*, DWH) je centrální, časově závislé úložiště historických i aktuálních dat. Jeho primárním účelem je podpora rozhodovacích procesů. Na rozdíl od provozních databází (OLTP) je struktura DWH optimalizována pro rychlé čtení, agregace a dotazování na velkých objemech dat.

Datové tržiště (*Data Mart*) je v korporátním prostředí definováno jako **podsložka datového skladu** zaměřená na data a metriky potřebné pro specifickou obchodní oblast (např. výroba, kvalita, prodej). Jedná se o menší, tématicky specializovanou entitu, která usnadňuje reportování a analýzu pro konkrétní skupinu uživatelů.

Konceptuální návrh DWH se opírá o dvě hlavní, avšak protichůdné, metodiky:

1. **Metodika Billa Inmona (*Top-Down Approach*):** Inmonova metodika je označována jako **Top-Down (shora dolů)**, protože začíná návrhem centrálního, podnikového datového skladu (*Enterprise Data Warehouse*, EDW).
    - **Schema:** EDW je modelováno ve vysoce **normalizované** formě (typicky 3. normální forma, 3NF), což zajišťuje nízkou datovou redundanci a maximální integritu.
    - **Tok dat a tržiště:** Data jsou nejprve ETL procesem načtena do detailního, normalizovaného EDW (centrální zdroj pravdy). **Datová tržiště** se vytváří **až sekundárně** z dat v EDW a jsou denormalizovaná, aby sloužila pro rychlé reportování.
2. **Metodika Ralpha Kimballa (*Bottom-Up Approach*):** Kimballova metodika je označována jako **Bottom-Up (zdola nahoru)**, protože se zaměřuje na rychlé dodání řešení pro specifické obchodní procesy.
    - **Schema:** Využívá **dimenzionální modelování** (schéma *Hvězda* nebo *Sněhová vločka*), které je záměrně **denormalizované**. To zjednodušuje dotazování a maximalizuje výkon pro OLAP úlohy.
    - **Tok dat a tržiště:** Data jsou transformována a ukládána **přímo** do dimenzionálních modelů, které **představují Datová tržiště**. Podnikový datový sklad je pak **logickou unií** (sjednocením) těchto jednotlivých tržišť.

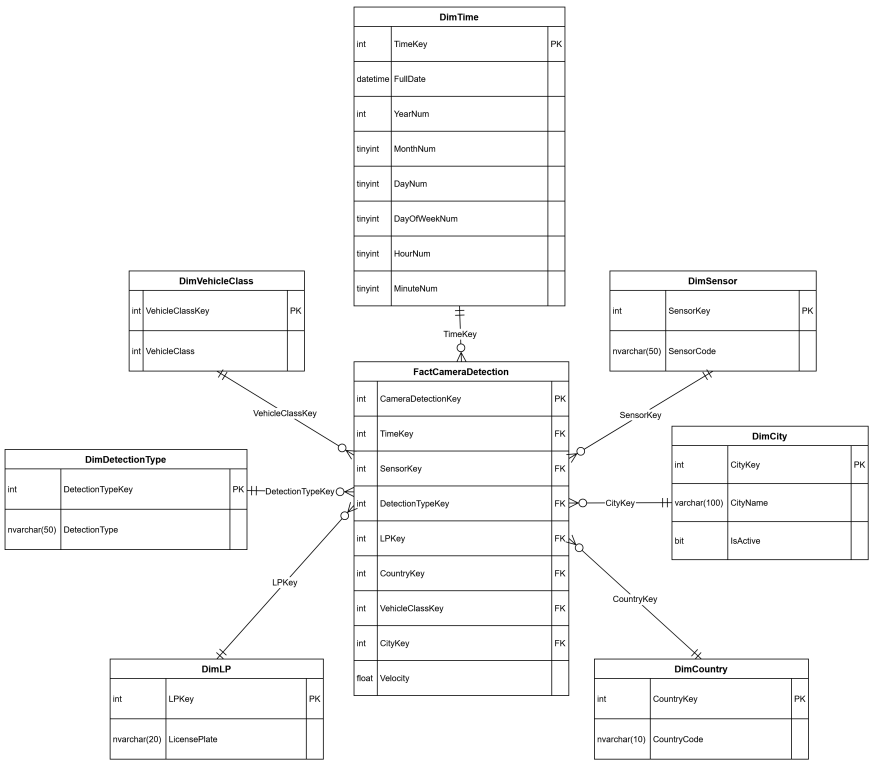
V praxi se však často objevují hybridní systémy kombinující prvky obou přístupů. Takovýto hybridní přístup jsem zvolil i já.

Data warehouse z pravidla obsahuje:

- **Staging Area:** Slouží jako dočasné úložiště, kam jsou data přenesena pomocí ETL (Extract, tansform and load). V této vrstvě se provádí **harmonizace a validace dat** před aplikací dimenzionálního modelu. Příkladem je tabulka `Stg.CameraCamea` ve Vaší implementaci (viz `bilina_kamery_lake_to_staging.py`).
- **Dimenze a Fakta:** Hlavní vrstva organizovaná dle Kimballova modelu (tedy Datový Mart pro tuto analytickou doménu). Skládá se z:
  1. **Dimenze (*Dimensions*):** Uchovávají kontext, atributy a popisné detaily (např. `DimCity`, `DimSensor`).
  2. **Fakta (*Facts*):** Uchovávají měřitelné hodnoty a cizí klíče k dimenzím (např. `FactCameraDetection`, viz `bilina_kamery_staging_to_fact.py`).

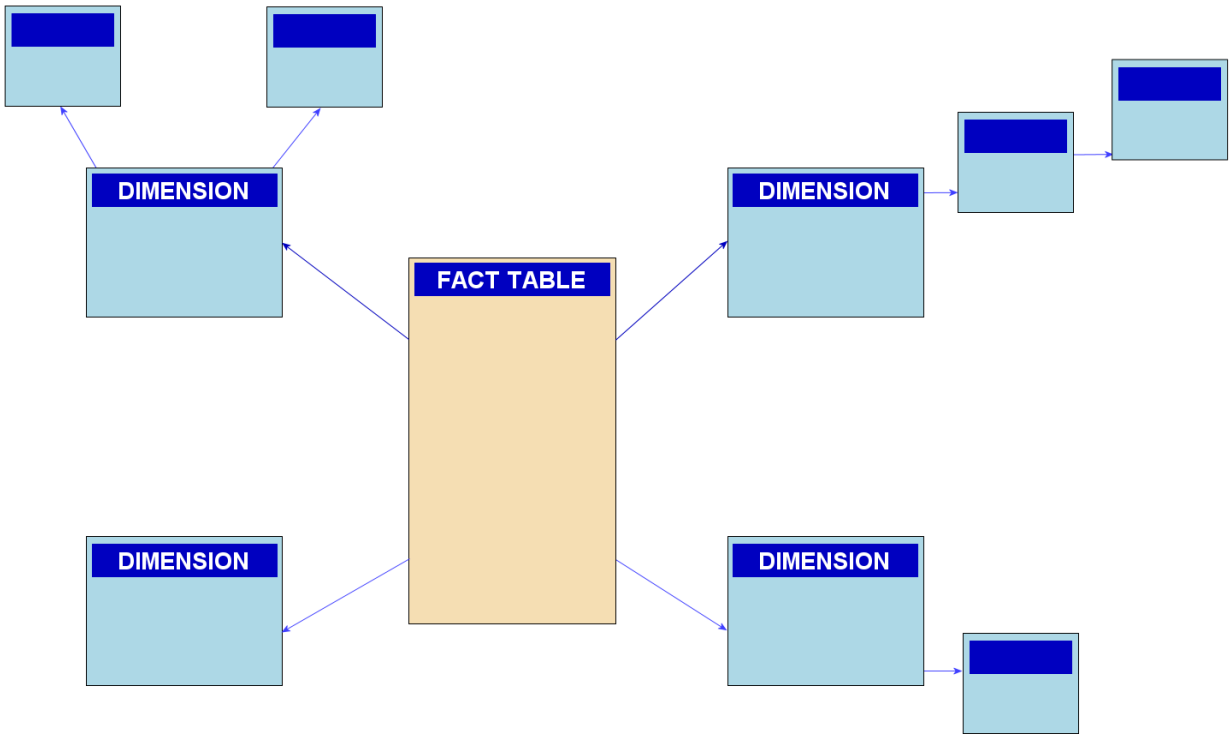
Základními modely používanými pro návrh datového skladu v rámci dimenzionálního modelování (Kimball) jsou:

- **Schéma Hvězda (*Star Schema*):** Jedná se o nejjednodušší a nejčastěji používaný dimenzionální model. Skládá se z centrální tabulky **Faktů** (*Fact Table*), která je obklopena několika tabulkami **Dimenzí** (*Dimension Tables*). Všechny dimenze jsou přímo napojeny na tabulku faktů, čímž vzniká struktura připomínající hvězdu. Vysoká redundance je vyvážena extrémní rychlostí dotazování.



Obrázek 2.2.: Schéma hvězda faktové tabulky a dimenzí pro data z kamer v Bílině.

- **Schéma Sněhová vločka (*Snowflake Schema*):** Jedná se o rozšíření schématu Hvězda, kde některé dimenze jsou **normalizovány** do několika souvisejících tabulek. Tím se snižuje redundance dat, ale na úkor zvýšení složitosti dotazování (je potřeba více `JOIN` operací), což může mírně zhoršit výkon.



Obrázek 2.3.: Ilustrační schéma sněhové vločky faktové tabulky a dimenzí převzato z [18].

OLAP technologie

OLAP (*Online Analytical Processing*) představuje přístup ke zpracování dat, který umožňuje provádět rychlé analytické dotazy nad velkými objemy informací. Základní myšlenkou OLAP je možnost pohledu na data z více dimenzí – typicky podle času, lokality, zařízení, typu senzoru nebo jiných analytických kritérií. Tento princip umožňuje uživatelům provádět tzv. *slice and dice* operace, tj. analyzovat data z různých perspektiv, agregovat je nebo filtrovat v reálném čase.

Rozlišují se tři tradiční typy OLAP řešení:

- **MOLAP (Multidimensional OLAP)** – pracuje s vlastní vícerozměrnou strukturou dat uloženou mimo relační databázi. Tento přístup poskytuje velmi rychlé odezvy při výpočtech a agregacích díky předpočítaným datovým strukturám (tzv. *cubes*), avšak vyžaduje rozsáhlejší přípravu a větší prostorové nároky.
- **ROLAP (Relational OLAP)** – využívá klasickou relační databázi, nad kterou jsou definovány pohledy a agregační dotazy. Tento přístup je flexibilnější, lépe škálovatelný a umožňuje přímou práci s aktuálními daty bez nutnosti jejich předpočítávání. Moderní nástroje, jako například **Microsoft SSAS Tabular** nebo **Apache Druid**, reprezentují evoluční formu

ROLAP – tzv. *in-memory tabulární modely*, které kombinují relační přístup se sloupcovým uložením dat a jazykem `DAX` pro definici metrik.

- **HOLAP (Hybrid OLAP)** – kombinuje výhody obou předchozích přístupů, tedy rychlé předpočítané agregace z MOLAP a flexibilitu dotazování nad detailními daty z ROLAP. Tento přístup je často využíván v moderních BI řešeních, kde se pro nejčastěji používané metriky udržují souhrnné cache.

Mezi nejpoužívanější nástroje pro implementaci OLAP vrstev patří například **Microsoft SQL Server Analysis Services (SSAS Tabular)** jako zástupce komerčního řešení, a z open-source prostředí pak zejména **Apache Kylin**, **Mondrian** (součást Pentaho) nebo **Cube.js**, který poskytuje moderní API pro analytické dotazy. Některé databázové systémy, jako například **ClickHouse**, navíc integrují funkce OLAP přímo do svého jádra, což umožňuje provádět agregace v reálném čase bez nutnosti vytvářet samostatnou analytickou vrstvu.

V současnosti je trendem přechod od klasických multidimenzionálních modelů k tabulárním řešením, která nabízejí vyšší flexibilitu, lepší integraci s nástroji pro vizualizaci dat a nižší nároky na údržbu datového modelu.

Sémantická vrstva

Sémantická vrstva (*Semantic Layer*) představuje logickou nadstavbu nad datovým skladem, jejímž hlavním cílem je sjednotit přístup k datům napříč celou organizací. Slouží jako prostředník mezi technickou strukturou databáze a uživatelskými nástroji Business Intelligence. Namísto přímé práce s databázovými tabulkami umožňuje uživatelům přistupovat k datům prostřednictvím předdefinovaných metrik, dimenzí a vztahů, které odpovídají obchodní logice organizace.

Tímto způsobem sémantická vrstva abstrahuje technické detaily a umožňuje vytváření analytických dotazů bez nutnosti znalosti SQL či fyzického modelu dat. V praxi tento koncept implementují moderní frameworky jako *Cube.js*, *dbt Semantic Layer* nebo *Looker Semantic Model*, které poskytují rozhraní pro standardizovaný přístup k datům prostřednictvím API nebo SQL proxy [3, 13, 19].

K hlavním přínosům sémantické vrstvy patří:

- **Konzistence metrik a výpočtů:** Veškeré reporty a dashboardy využívají jednotně definované ukazatele, čímž se eliminuje riziko rozdílné interpretace dat.
- **Zjednodušení analytické práce:** Uživatelé mohou tvořit vizualizace nebo reporty bez nutnosti znalosti komplexní struktury DWH.
- **Zvýšení výkonu a bezpečnosti:** Sémantická vrstva často zajišťuje cachování a řízení přístupů na úrovni obchodních objektů, což zlepšuje odezvu systému a kontrolu nad daty.

Z pohledu architektury datových platforem tak sémantická vrstva představuje klíčový prvek mezi datovým skladem a vizualizační vrstvou, který propojuje technologickou přesnost s obchodní srozumitelností.

2.2. Nástroje Business Intelligence

Nástroje Business Intelligence (BI) představují softwarové řešení určené k transformaci surových dat na informace, které podporují rozhodovací procesy na všech úrovních řízení. Cílem BI systémů je zajistit uživatelům přístup k aktuálním, konzistentním a relevantním datům, a to formou interaktivních vizualizací, reportů či analytických přehledů. V rámci moderních datových platforem, jako je Portabo, tvoří BI nadstavbu nad datovým skladem a OLAP vrstvou. Zprostředkovává uživatelům přístup k datům v srozumitelné a vizuálně přitažlivé podobě a umožňuje jejich interpretaci bez nutnosti znalosti technických detailů implementace.

BI nástroje dnes představují klíčový prvek datově řízeného rozhodování a zahrnují široké spektrum funkcí – od základních přehledů a dashboardů až po pokročilé analytické a prediktivní modely využívající strojové učení.

Funkční oblasti nástrojů Business Intelligence

Nástroje BI lze z hlediska jejich zaměření rozdělit do několika funkčních oblastí, které společně pokrývají celý proces přeměny dat na znalosti:

- **Reportování a vizualizace dat** – umožňuje prezentaci dat formou tabulek, grafů a interaktivních dashboardů. Uživatelé tak mohou rychle identifikovat klíčové ukazatele výkonnosti (KPI), sledovat trendy a vyhodnocovat vývoj v čase.
- **Ad-hoc analýza a průzkum dat** – nabízí uživatelům možnost vytvářet vlastní pohledy na data, provádět filtrování, seskupování nebo detailní analýzy (tzv. *drill-down* a *drill-up*). Tato oblast je důležitá pro datové analytiky a business specialisty, kteří potřebují flexibilně reagovat na nové otázky bez zásahu IT oddělení.
- **Pokročilá analytika a prediktivní modelování** – integruje BI s nástroji pro datovou vědu a strojové učení. Umožňuje například předpověď trendů, klasifikaci nebo detekci anomálií. Tyto funkce bývají častější u komerčních řešení, která využívají propojení s cloudovými službami nebo integrované modelovací prostředí (např. Microsoft Azure Machine Learning).
- **Sdílení a spolupráce** – moderní BI nástroje podporují publikaci reportů a dashboardů napříč organizací, správu oprávnění, exporty a integraci s komunikačními nástroji (např. Microsoft Teams, Slack či SharePoint). Tato funkce je zásadní pro efektivní distribuci informací mezi jednotlivé úrovně řízení.

Přehled dostupných BI nástrojů

Z pohledu dostupných řešení lze nástroje Business Intelligence rozdělit do dvou hlavních kategorií – open-source a komerčních platforem. Obě skupiny mají své výhody a nevýhody, které je nutné zohlednit při rozhodování o jejich implementaci v konkrétním prostředí.

- **Open-source řešení** – nástroje jako *Grafana*, *Apache Superset*, *Metabase* nebo *Redash* poskytují široké možnosti integrace s relačními i nestrukturovanými datovými zdroji. Jsou vhodné zejména pro organizace, které preferují flexibilitu, nižší licenční náklady a možnost přizpůsobit systém vlastním potřebám. Nevýhodou bývá nutnost technické správy, složitější počáteční konfigurace a omezená úroveň uživatelské podpory.
- **Komerční řešení** – mezi nejrozšířenější komerční platformy patří *Microsoft Power BI*, *Tableau* a *Qlik Sense*. Tyto nástroje se vyznačují vysokou mírou integrace s podnikovými systémy, širokými možnostmi automatizace a pokročilými funkcemi pro datové modelování, sdílení a prediktivní analýzy. Výhodou bývá profesionální podpora výrobce, pravidelné aktualizace a integrace s cloudovými službami. Na druhé straně představují komerční nástroje vyšší finanční náklady a častou závislost na dodavateli (tzv. *vendor lock-in*).

Kritéria výběru BI nástroje

Volba vhodného BI nástroje závisí na potřebách organizace, technické infrastruktuře a rozpočtu. Při rozhodování je vhodné zohlednit následující kritéria:

1. **Uživatelská přívětivost a intuitivní rozhraní** – nástroj by měl umožnit tvorbu reportů i netechnickým uživatelům bez nutnosti psaní SQL dotazů.
2. **Konektivita k datovým zdrojům** – podpora přímého připojení k datovému skladu, OLAP vrstvám či externím API.
3. **Automatizace a aktualizace dat** – schopnost plánovat obnovu dat, vytvářet datové toky (pipelines) a spravovat přístupová oprávnění.
4. **Možnosti spolupráce a sdílení** – integrace s podnikovými systémy (např. Microsoft Teams, Slack, SharePoint) a možnost publikování interaktivních dashboardů.
5. **Licenční model a celkové náklady na provoz (TCO)** – kromě pořizovací ceny je třeba zohlednit náklady na údržbu, školení uživatelů a případnou infrastrukturu.

V praxi organizace často kombinují více nástrojů – například open-source řešení pro interní analýzy a komerční systém pro prezentaci výsledků managementu. Tento přístup umožňuje optimalizovat náklady a zároveň využít silných stránek jednotlivých platforem. V případě datové platformy Portabo by mohl být takový přístup vhodný při implementaci vizualizační vrstvy nad datovým skladem.

2.3. Přehled a charakteristika využitých technologií

Pro účely praktické části a srovnávací analýzy byl vybrán reprezentativní soubor technologií, které pokrývají jak komerční, tak open-source přístupy k budování datové platformy. Následující tabulka poskytuje jejich přehled, charakteristiku a popisuje jejich specifickou roli v navržené architektuře.

Open-source řešení	Komerční řešení
Vrstva: Datový sklad	
PostgreSQL a TimescaleDB PostgreSQL je open-source relační databázový systém s podporou SQL standardu. Pro potřeby této práce je klíčové jeho rozšíření **TimescaleDB**, které PostgreSQL transformuje na vysoce výkonnou databázi pro časové řady. Tato kombinace je vhodná pro telemetrická data, jako jsou data ze senzorů a kamer, díky optimalizacím pro časově orientované dotazy a efektivní kompresi dat. **ClickHouse** ClickHouse je open-source, sloupcově orientovaný databázový systém navržený primárně pro online analytické zpracování (OLAP). Jeho architektura je optimalizována pro extrémně rychlé provádění agregačních dotazů nad velkými objemy dat. V rámci srovnání slouží jako zástupce vysoce výkonných analytických databází nové generace, které umožňují analýzu dat v reálném čase bez nutnosti rozsáhlých předagregací. **MariaDB** MariaDB je open-source relační databázový systém, který vznikl jako fork MySQL a zachovává si s ním vysokou kompatibilitu. Pro potřeby této práce je klíčové jeho rozšíření **MariaDB ColumnStore**, které transformuje tradiční řádkově orientovanou databázi na hybridní systém schopný sloupcového ukládání dat. Tato architektura je optimalizována pro analytické dotazy (OLAP)	**Microsoft SQL Server** Microsoft SQL Server (MSSQL) je komplexní komerční systém pro správu relačních databází (RDBMS). Kromě standardních transakčních (OLTP) operací nabízí robustní nástroje pro datové sklady, včetně integračních služeb (SSIS) a analytických služeb (SSAS). V této práci je MSSQL využit jako jeden z pilířů pro testování komerčního řešení, konkrétně jako databázový engine pro datový sklad a zároveň jako platforma pro SSAS Tabular model.

Open-source řešení	Komerční řešení
Vrstva: Sémantická vrstva / OLAP	
Cube.js Cube.js je open-source sémantická vrstva (semantic layer), jejímž hlavním úkolem je abstrahovat technickou komplexitu datového skladu a poskytovat konzistentní business logiku. Definuje datové modely, metriky (measures) a dimenze pomocí JavaScriptu. Klíčovou vlastností je poskytování standardizovaného rozhraní (API), včetně emulace **PostgreSQL SQL API**. To umožňuje, aby se vizualizační nástroje (jako Apache Superset) připojovaly ke Cube.js stejným způsobem, jako by se připojovaly k běžné databázi, čímž se výrazně zjednodušuje integrace a zvyšuje výkon díky pokročilému cachování a předagregacím.	**Microsoft SQL Server Analysis Services** SSAS je komponenta Microsoft SQL Serveru určená pro OLAP a Business Intelligence. V této práci je využíván jeho **Tabulární model (Tabular Model)**, který funguje jako *in-memory* databáze optimalizovaná pro analytické dotazy pomocí jazyka `DAX` (Data Analysis Expressions). Slouží jako komerční protějšek k Cube.js, poskytující podobnou sémantickou vrstvu s pokročilými možnostmi bezpečnosti, správy a hlubokou integrací s nástroji Microsoft ekosystému.
Vrstva: Vizualizace / BI	
Apache Superset Apache Superset je moderní open-source platforma pro vizualizaci dat a business intelligence. Umožňuje snadné připojení k široké škále datových zdrojů (včetně SQL API poskytovaného Cube.js) a intuitivní tvorbu interaktivních dashboardů a reportů. V této práci reprezentuje flexibilní open-source řešení pro vizualizační vrstvu, které klade důraz na samoobslužnou analytiku a customizaci.	**Microsoft Power BI** Power BI je komerční analytický nástroj od společnosti Microsoft, který představuje průmyslový standard pro vizualizaci dat a business intelligence. Vyniká těsnou integrací s celým ekosystémem Microsoftu, zejména s SSAS Tabular, což umožňuje efektivní analýzu, sdílení reportů a spolupráci v rámci organizace. V kontextu této práce slouží jako reprezentant komerční vizualizační platformy s pokročilými funkcemi pro datovou transformaci, modelování a distribuci obsahu.

3. Praktická část

Praktická část této práce se zaměřuje na návrh, implementaci a ověření metodiky pro srovnání open-source a komerčních nástrojů využitelných pro analýzu a vizualizaci dat na datové platformě Portabo. Cílem bylo vytvořit plně funkční datový sklad s podporou OLAP analýz, který umožňuje zpracovávat reálná data poskytovaná datovým centrem Ústeckého kraje a připravit prostředí pro následné výkonové a funkční porovnání vizualizačních nástrojů.

V rámci řešení byly navrženy a implementovány datové toky, které respektují jednotnou architekturu systému složeného z vrstev *datového jezera*, *datového skladu* (staging vrstva, dimenze, fakta), *OLAP vrstvy* a *reportingu*. Tato struktura byla zachována napříč všemi testovanými technologiemi, přičemž pro jednotlivé implementace byl použit stejný princip ETL zpracování, stejný způsob transformace a obdobné rozvržení datového modelu založeného na dimenzionálním schématu.

Data využitá pro testování pocházejí převážně z datového toku města Bílina, konkrétně z MQTT témat obsahujících telemetrické údaje o kamerových detekcích vozidel. V případě implementace nad PostgreSQL/TimescaleDB byla navíc použita širší množina dat zahrnující také Wi-Fi senzory, elektrické měřiče a další zdroje. Tyto zprávy byly zpracovány pomocí implementovaných ETL procesů a uloženy do datového skladu vytvořeného nad několika databázovými technologiemi (PostgreSQL/TimescaleDB, MariaDB, Microsoft SQL Server a ClickHouse). Každá z těchto technologií představuje odlišný přístup – od komerčního řešení po výkonné open-source systémy určené pro analytické zpracování dat.

Příprava a migrace datové základny

Výchozím bodem pro naplnění datového jezera (Data Lake), které sloužilo jako zdroj pro všechny následné ETL procesy, byl poskytnutý SQL dump určený pro databázi **MariaDB**. Jelikož metodika srovnání zahrnovala i jiné databázové systémy, bylo nutné provést migraci těchto dat do jednotlivých cílových prostředí. Pro každou technologii byl zvolen specifický přístup:

Microsoft SQL Server: Pro migraci do MSSQL byl zvolen dvoukrokový přístup, neboť původní pokusy o vytvoření vlastního konverzního skriptu se ukázaly jako nespolehlivé.

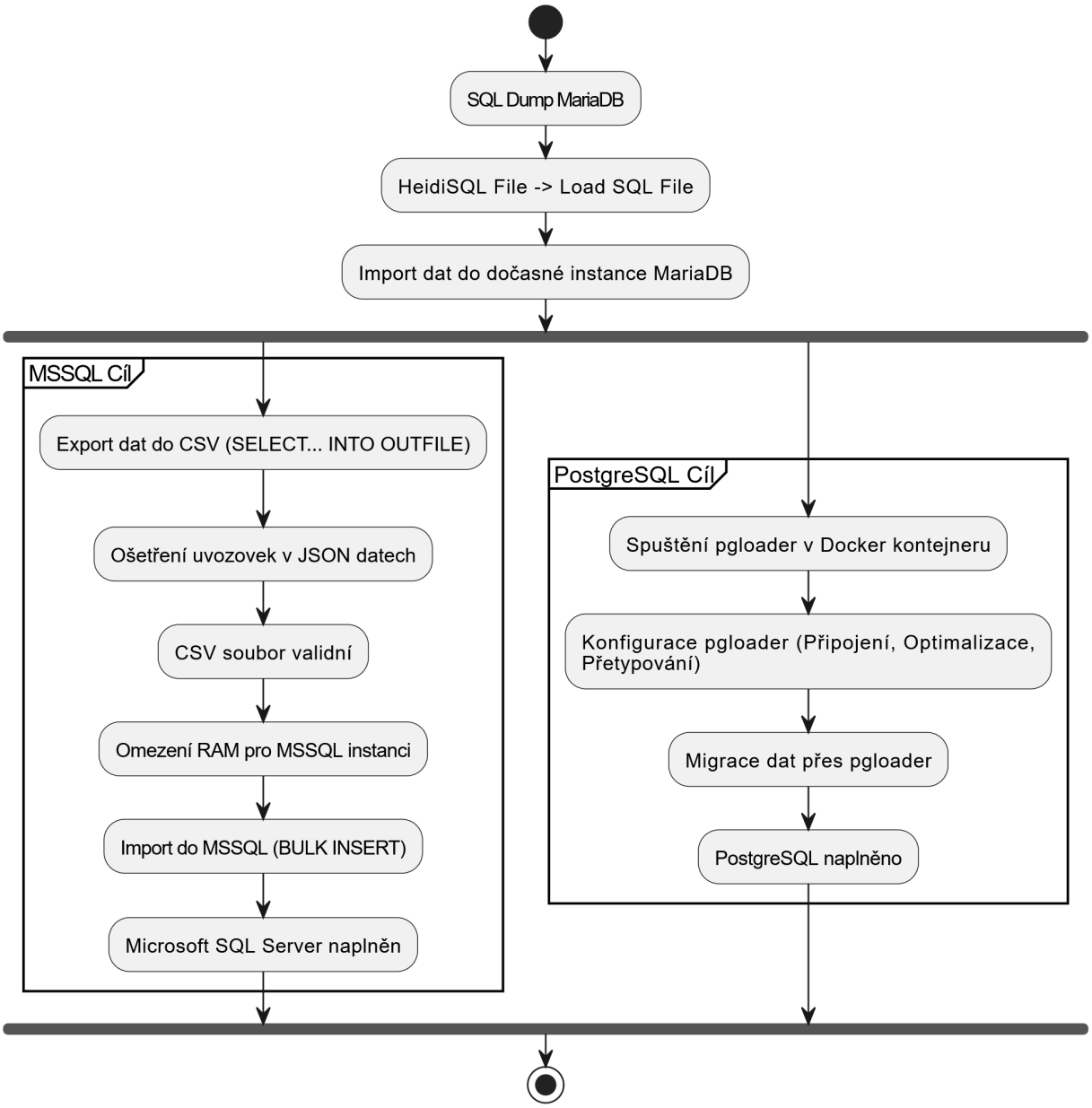
1. **Export z MariaDB:** Data byla nejprve naimportována do dočasné instance MariaDB. Následně byla z této databáze exportována do univerzálního formátu CSV pomocí SQL dotazu `SELECT ... INTO OUTFILE`. Tento export vyžadoval specifické ošetření uvozovek v JSON datech, aby byl výsledný soubor validní.

2. **Import do MSSQL:** Výsledný CSV soubor byl naimportován do MSSQL pomocí příkazu `BULK INSERT`. Během tohoto procesu bylo také nutné manuálně omezit maximální přidělenou operační paměť (RAM) pro instanci MSSQL serveru, aby nedošlo k zahlcení systémových prostředků.

PostgreSQL: Pro migraci do PostgreSQL byl využit nástroj `pgloader`, který umožňuje přímou migraci mezi různými databázovými systémy.

- **Konfigurace a spuštění:** Nástroj byl spuštěn v rámci vlastního Docker kontejneru. Konfigurační soubor definoval přímé připojení ke zdrojové MariaDB databázi (`host.docker.internal:3306`) a cílové PostgreSQL databázi (`host.docker.internal:5433`).
- **Optimalizace výkonu:** Konfigurace `pgloader` byla optimalizována pro co nejvyšší rychlost přenosu. Bylo využito paralelní zpracování (`workers = 4`, `concurrency = 4`), dávkování dat po 50 000 řádcích (`batch rows = 50000`) a navýšení operační paměti pro proces migrace (`maintenance_work_mem to '512MB'`).
- **Přetypování dat:** Součástí konfiguračního souboru byla také explicitní pravidla pro přetypování datových typů mezi oběma systémy, zejména pro správnou interpretaci časových údajů (`CAST type datetime to timestamptz...`).

Těmito postupy bylo zajištěno, že identická datová základna (datové jezero) byla k dispozici ve všech testovaných databázových systémech.



Obrázek 3.1.: Aktivity diagram nahrání MariaSQL dumpu do odlišných databází.

3.1. Návrh metodiky porovnání

Cílem metodiky porovnání je zajistit objektivní a opakovatelné vyhodnocení open-source a komerčních technologií využitelných pro analýzu a vizualizaci dat v prostředí platformy Portabo. Porovnání je založeno na principech jednotného datového toku, jednotného datového modelu a shodného způsobu zpracování dat. Tím je zaručeno, že rozdíly ve výsledcích budou dány samotnými vlastnostmi testovaných technologií, nikoli odlišnostmi v implementaci.

Každá z porovnávaných variant bude hodnocena z následujících hledisek:

- **Výkonové parametry** – rychlost načítání a zpracování dat, doba odezvy analytických dotazů, efektivita indexace a agregací;
- **Funkční vlastnosti** – podpora OLAP analýz, kompatibilita s vizualizační vrstvou a možnosti rozšiřitelnosti;
- **Uživatelská přívětivost** – náročnost správy, dostupnost nástrojů a srozumitelnost konfigurace;
- **Ekonomické aspekty** – licenční politika, náklady na provoz, údržbu a škálování.

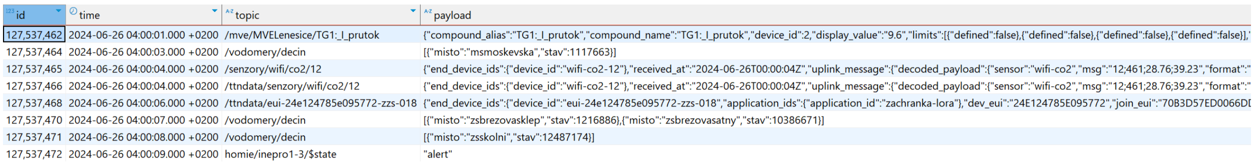
Metodika tedy určuje rámec, v němž budou jednotlivé technologie nasazeny, naplněny shodnými daty a následně testovány.

3.2. Příprava dat a návrh datového skladu

Analýza a příprava zdrojových dat

V úvodní fázi implementace byla vybrána konkrétní datová doména, nad kterou bude systém postaven. V tomto případě se jedná o kamerový systém města Bílina. Cílem bylo ověřit, zda je možné tato data zpracovávat a následně připravit půdu pro jejich dynamické zpracování.

Poskytnutý `MariaDB` SQL dump obsahoval následující sloupce: `id (bigint(11))`, `time (timestamp)`, `topic (tinytext)` a `payload (text)`. Tyto údaje představovaly surová telemetrická data, původně pocházející z MQTT komunikace, která byla exportována do databázového dumpu. Tento dump sloužil jako vstupní datová sada pro analýzu a testování datového toku.



id	time	topic	payload
127,537,462	2024-06-26 04:00:01.000 +0200	/mve/MVELenesice/TG1:_I_prutok	{"compound_alias":"TG1:_I_prutok","compound_name":"TG1:_I_prutok","device_id":2,"display_value":"9.6","limits":[{"defined":false},{"defined":false},{"defined":false},{"defined":false}],'
127,537,464	2024-06-26 04:00:03.000 +0200	/vodomery/decin	[{"misto":"msmoskevska","stav":1117663}]
127,537,465	2024-06-26 04:00:04.000 +0200	/senzory/wifi/co2/12	{"end_device_ids":{"device_id":"wifi-co2-12"},"received_at":"2024-06-26T00:00:04Z","uplink_message":{"decoded_payload":{"sensor":"wifi-co2","msg":"12;461;28.76;39.23","format":"
127,537,466	2024-06-26 04:00:04.000 +0200	/ttndata/senzory/wifi/co2/12	{"end_device_ids":{"device_id":"wifi-co2-12"},"received_at":"2024-06-26T00:00:04Z","uplink_message":{"decoded_payload":{"sensor":"wifi-co2","msg":"12;461;28.76;39.23","format":"
127,537,468	2024-06-26 04:00:06.000 +0200	/ttndata/eui-24e124785e095772-zzs-018	{"end_device_ids":{"device_id":"eui-24e124785e095772-zzs-018","application_ids":{"application_id":"zachranka-lora"},"dev_eui":"24E124785E095772","join_eui":"70B3D57ED0066DI
127,537,470	2024-06-26 04:00:07.000 +0200	/vodomery/decin	[{"misto":"zsbrezovasklep","stav":1216886},{"misto":"zsbrezovasatny","stav":10386671}]
127,537,471	2024-06-26 04:00:08.000 +0200	/vodomery/decin	[{"misto":"zsskolni","stav":12487174}]
127,537,472	2024-06-26 04:00:09.000 +0200	homie/inepro1-3/$state	"alert"

Obrázek 3.2.: Struktura tabulky `mqttentries` obsahující zdrojová telemetrická data

Z obrázku je patrná struktura vstupních dat. Ze sloupce `topic` lze částečně odvodit, o jaký typ senzoru se jedná. Podrobnější analýza však ukázala, že názvy témat nejsou konzistentní. Například

kamera z Bíliny může mít téma `/Bilina/kamery/camea/BI-TP-02`, zatímco vodoměr je označen jednoduše `/vodomery/decin`. V databázi se rovněž vyskytují senzory, které vykonávají stejnou funkci, avšak používají odlišné pojmenování.

Z těchto důvodů bylo vyhodnoceno, že vytvoření skriptu, který by dynamicky parsoval JSON struktury na základě názvů témat, není vzhledem k nekonzistenci dat reálně možné. Bylo nutné zvolit robustnější přístup založený na analýze obsahu samotných zpráv.

Automatizovaná analýza JSON struktur

Pro systematickou analýzu JSON struktur byl vytvořen vlastní skript v jazyce Python, `analyze_json.py`. Tento skript seskupuje MQTT témata do logických celků na základě strukturální podobnosti jejich JSON schémat ve sloupci `payload`.

K porovnání struktury dvou JSON objektů byla využita **Jaccardova podobnost**, která měří poměr velikosti průniku a sjednocení množin jejich klíčů. Analýza s prahovou hodnotou 100 % shody odhalila větší množství identických struktur, avšak některé senzory se lišily pouze v několika atributech – pravděpodobně kvůli odlišným verzím firmwaru. Proto byla hodnota podobnosti postupně snižována až na 50 %, čímž se podařilo identifikovat širší spektrum vzájemně příbuzných JSON struktur.

Výstupem této první fáze analýzy jsou skupiny témat, které sdílejí podobnou datovou strukturu, jak je znázorněno na obrázku 3.3.

GroupID	NumTopics	Topics	Keys
1	31	/Bilina/kamery/camea/BI-MO-I1, /Bilina/kamery/camea/BI-MO-I1B, /Bilina/kamery/camea/BI-MO	detectiontype:object:value, ilpc:object:value, lp:object:value, sensor:object:value, utc:object:value, v
5	2	/detektory/video/brna/brnaJF, /detektory/video/brna/cyklotrasa	format:object:value, msg:object_countdata:object_data:object_array:scitani:object_left _> right:obj
7	1254	/Energo/DCUK/INEPRO-Pro380-Mod/inepro1-2/ApparentPower, /Energo/DCUK/INEPRO-Pro380-M	value:value
17	3	/mve/MVELibochovice, /mve/libochovice, /mve/patek	gen1vykcin:object_array:value, gen2vykcin:object_array:value, hlajezmnm:object_array:value, hlapo
18	3	/mve/MVELenesice/NS:_DI_hl_horni, /mve/MVELenesice/TG1:_I_prutok, /mve/MVELenesice/TG2:	alias:object:value, archived:object:value, compound_alias:object:value, compound_name:object:valu
21	12	/senzory/wifi/co2/11, /senzory/wifi/co2/12, /senzory/wifi/co2/13, /senzory/wifi/co2/14, /senzory/	end_device_ids:object_device_id:object:value, received_at:object:value, uplink_message:object_dec
22	288	/ttndata/eui-24e124136c489089-02, /ttndata/eui-24e124136c489130-03, /ttndata/eui-24e124136c	correlation_ids:object_array:value, end_device_ids:object_application_ids:object_application_id:obj
25	14	/ttndata/voda/eui-24e124713d167798, /ttndata/voda/eui-24e124713d321242-vzdalenost-06, /ttnc	app_id:object:value, battery:object:value, dev_id:object:value, distance:object:value, time:object:val
26	169	/udp1881/15102001, /udp1881/15102002, /udp1881/15102004, /udp1881/15102005, /udp1881/15	_msgid:object:value, _session:object_id:object:value, _session:object_type:object:value, battery:obj

Obrázek 3.3.: Prvotní seskupení témat na základě 50 % strukturální podobnosti JSON schémat.

V dalším kroku bylo nutné tyto automaticky vygenerované skupiny ručně zrevidovat. Ukázalo se, že některé skupiny, ačkoliv měly mírně odlišnou JSON strukturu, patřily ke stejnému typu zařízení, které pouze měřilo jinou veličinu (např. jeden senzor měřící napětí a druhý proud). Tyto logicky související skupiny byly následně sloučeny do finálních celků určených pro ETL zpracování. Tento proces je znázorněn na obrázku 3.4.

GroupID	NumTopics	Topics	Keys
8	2	/Energo/DCUK/SML133/SML133-01, /Energo/DCUK/SML133/SML133-01/act	subjson:object__3i:object:value, subjson:object__cos1:object:value, subjson:object__cos2:object:val
10	2	/Energo/DCUK/SML133/SML133-01/har, /Energo/DCUK/SML133/SML133-01/osc	subjson:object__delta:object:value, subjson:object__device:object:value, subjson:object__fi1:object

Obrázek 3.4.: Tyto logické skupiny byly navíc ještě manuálně sloučeny napříč skupiny

Návrh a principy ETL procesů

Na základě analýzy dat byly navrženy a implementovány samostatné ETL skripty pro jednotlivé databázové technologie. Každý z těchto skriptů zajišťuje přenos dat z *datového jezera* (surová MQTT data) do *staging vrstvy datového skladu*, kde jsou data očištěna a standardizována. Následně jsou zpracovaná data přesunuta do faktové tabulky `FactCameraDetection`, která tvoří jádro analytické vrstvy systému.

Všechny implementace sdílejí jednotný princip inkrementálního načítání dat. Každý běh ETL skriptu začíná načtením posledního zpracovaného identifikátoru z tabulky `ETL_IncrementalControl` (sloupec `LastLoadedID`). Tímto způsobem se při každém dalším běhu zpracovávají pouze nově přijaté zprávy. Průběh každého běhu je zapisován do tabulky `ETL_RunLog`, kde jsou evidovány údaje jako název úlohy, čas spuštění a stav (`RUNNING`, `SUCCESS` či `FAILED`).

Data jsou zpracovávána v dávkách (tzv. *batchích*), jejichž velikost je řízena parametrem `BATCH_SIZE`. Každý skript používá databázové kurzory a příkazy `fetchmany()`, aby se minimalizovalo zatížení paměti. Klíčovým polem v těchto datech je `payload`, obsahující JSON strukturu s detaily detekce. Pro jeho zpracování byly v rámci práce implementovány dva odlišné přístupy:

- **Statické ETL (SQL Server, MariaDB, MariaDB+ClickHouse):** Tento přístup byl implementován pro data z kamerového systému města Bílina (`/Bilina/kamery/camea/%`). Vytvořené skripty využívají předem definovanou znalost struktury JSON dat a atributy z `payload` jsou mapovány na pevně dané sloupce ve staging tabulce.
- **Dynamické ETL (PostgreSQL s TimescaleDB):** Naopak pro implementaci nad databází PostgreSQL byl zvolen univerzální přístup. Skript (`pg_timescale_lake_to_staging.py`) byl navržen tak, aby se dokázal automaticky adaptovat na různorodé JSON struktury. Během zpracování dynamicky analyzuje obsah `payload`, odvozuje datové typy a v případě potřeby sám upravuje cílovou staging tabulku příkazem `ALTER TABLE`.

Při zpracování dat jsou prováděny převody datových formátů, ošetřovány chybějící hodnoty a celý proces je obalen v transakcích a blocích `try/except` pro zajištění konzistence a logování chyb.

Implementace ETL pro jednotlivé technologie

SQL Server ETL

ETL proces pro SQL Server je dvoukrokový a obsluhují jej dva samostatné skripty.

- **Krok 1: Z datového jezera do stagingu (`mssql_bilina_kamery_lake_to_staging.py`)**
  Tento skript využívá knihovnu `pyodbc`. Inkrementálně načítá surová data z tabulky `mqttentries`, parsuje JSON `payload` a očištěné záznamy vkládá po dávkách do staging tabulky `[Stg].[CameraCamea]`. Každý běh je logován v tabulce `ETL_RunLog`, přičemž pro získání ID běhu využívá T-SQL klauzuli `OUTPUT inserted.RunID`.

- **Krok 2: Ze stagingu do dimenzí a faktů (`mssql_bilina_kamery_staging_to_fact`)** Druhý skript zpracovává data připravená ve stagingu. Pro nalezení nových dimenzních hodnot nejprve vloží unikátní záznamy z dávky do dočasné tabulky `##TempDimensions`. Následně pomocí T-SQL klauzule `EXCEPT` porovná obsah dočasné tabulky s existujícími dimenzemi a vloží pouze chybějící záznamy. Poté konstruuje záznamy pro faktovou tabulku `FactCameraDetection` pomocí `LEFT JOIN` na dimenze.

MariaDB + MariaDB ColumnStore ETL

Implementace pro MariaDB se řídí stejnou dvoukrokovou logikou, avšak krok 2 je přizpůsoben pro specifika enginu ColumnStore.

- **Krok 1: Z datového jezera do stagingu (`maria_bilina_kamery_lake_to_staging.py`)** Skript používá knihovnu `pymysql`. Načítá data z datového jezera, zpracovává JSON a ukládá je do staging tabulky `Stg_CameraCamea`. Pro aktualizaci kontrolní tabulky využívá MySQL klauzuli `ON DUPLICATE KEY UPDATE`.
- **Krok 2: Ze stagingu do dimenzí a faktů (`maria_bilina_kamery_staging_to_fact.py`)** Tento skript používá pro plnění dimenzí příkaz `INSERT IGNORE`, který přeskočí již existující záznamy. Při plnění faktové tabulky skript nejprve v Pythonu načte všechny dimenzní klíče do paměti, následně projde staging data, nahradí původní hodnoty číselnými klíči a celý výsledek zapíše do dočasného CSV souboru. Nakonec je tento soubor nahrán do faktové tabulky pomocí příkazu `LOAD DATA LOCAL INFILE`. Protože MariaDB v enginu ColumnStore nepodporuje auto-inkrementaci, je generování primárního klíče řešeno přímo v ETL skriptu.

PostgreSQL + PostgreSQL TimescaleDB ETL

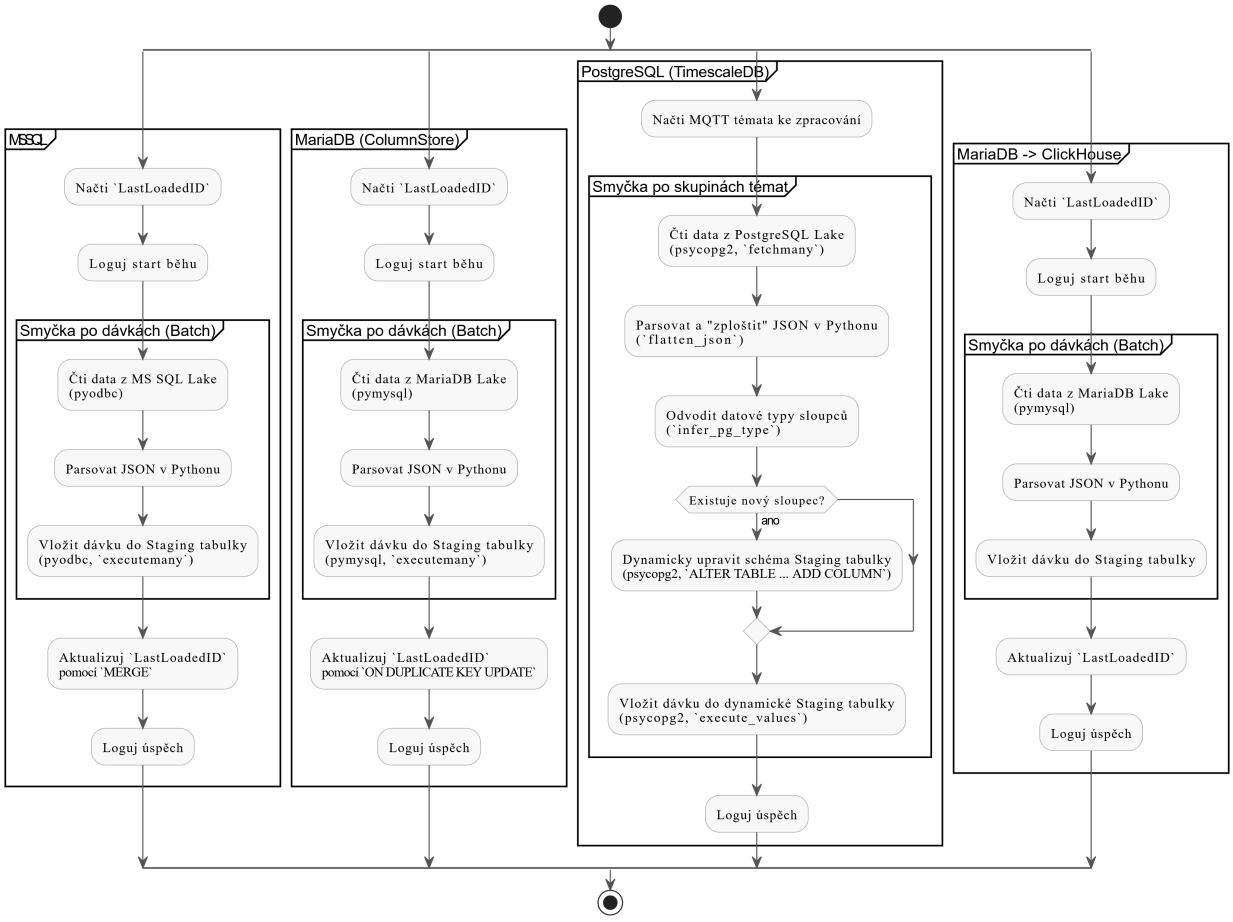
ETL proces pro PostgreSQL je unikátní svým dynamickým přístupem v prvním kroku a hromadným vkládáním ve druhém.

- **Krok 1: Dynamické ETL z jezera do stagingu (`pg_timescale_lake_to_staging.py`)** Tento skript automaticky analyzuje JSON `payload`. Pomocí funkcí `flatten_json()` a `infer_pg_type` dynamicky odvozuje schéma a datové typy. Pokud narazí na nový atribut, sám upraví cílovou staging tabulku příkazem `ALTER TABLE`.
- **Krok 2: Ze stagingu do dimenzí a faktů (`pg_timescale_staging_to_fact.py`)** Druhý skript využívá pro plnění dimenzí specifickou vlastnost PostgreSQL – příkaz `INSERT ... ON CONFLICT DO NOTHING`. Tento příkaz je spouštěn pomocí optimalizační funkce `execute_values` z knihovny `psycopg2`, která sestaví jediný rozsáhlý `INSERT` příkaz obsahující všechny hodnoty najednou. Toto umožňuje databázi zpracovat celou dávku v jediné operaci. Následně jsou data ze stagingu pomocí standardního `INSERT INTO ... SELECT` s `LEFT JOIN`y transformována do faktové tabulky.

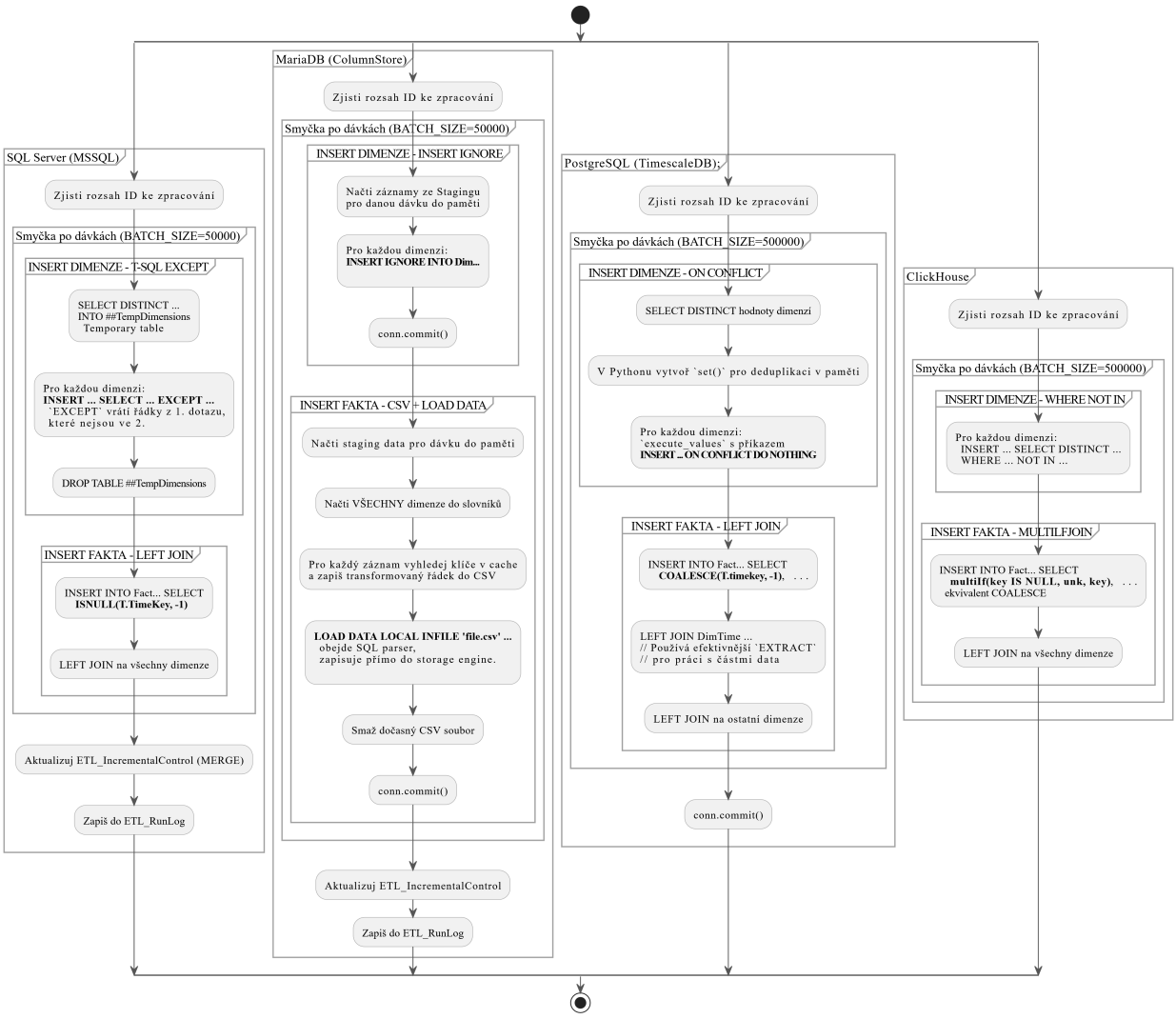
MariaDB + ClickHouse ETL

Tato varianta se odlišuje přímým datovým mostem mezi dvěma různými technologiemi a specifickým způsobem přenosu dat.

- **Krok 1: Z datového jezera do stagingu (`maria_click_bilina_kamery_lake_to_staging.py`)** Skript zajišťuje přenos dat z MariaDB do ClickHouse. Pro zápis do ClickHouse je využit nativní binární protokol (port 9000), přičemž data jsou v Pythonu agregována do dávky záznamů (tuples) a odeslána v jedné operaci. Data z MariaDB jsou čtena proudově (streaming) po menších dávkách, čímž se zamezuje načtení celého objemu dat do paměti najednou. Veškeré řídicí a logovací tabulky (`ETL_IncrementalControl`, `ETL_RunLog`) jsou spravovány přímo v ClickHouse.
- **Krok 2: Ze stagingu do dimenzí a faktů (`maria_click_kamery_staging_to_fact.py`)** Celý tento krok probíhá výhradně v rámci ClickHouse. Skript spouští sérii SQL dotazů. Pro naplnění dimenzí se používá konstrukce `INSERT INTO ... SELECT ... WHERE NOT IN`, která vybere a vloží pouze nové hodnoty. Následně jsou data transformována do faktové tabulky jediným příkazem `INSERT INTO ... SELECT` obsahujícím všechny potřebné `LEFT JOIN`y na dimenzní tabulky.



Obrázek 3.5.: UML porovnání ETL scriptů napříč systémy.



Obrázek 3.6.: UML porovnání ETL scriptů napříč systémy.

stgid	landingid	originaltime	topic	loaddttm	velocity	ilpc	vehclass	sensor	lp	utc	detectiontype
4	127,537,482	2024-06-26 04:00:12.000 +0200	/Bilina/kamery/camea/BI-TP-I2	2025-10-26 15:04:33.922 +0100	-1	CZ	0	BI-TP-I2	EDAE444B79	2024-06-25 09:27:20.729 +0200	detector
5	127,537,483	2024-06-26 04:00:12.000 +0200	/Bilina/kamery/camea/BI-TP-O2B	2025-10-26 15:04:33.922 +0100	-1	CZ	0	BI-TP-O2B	97746D2A69	2024-06-25 09:27:20.951 +0200	detector
6	127,537,484	2024-06-26 04:00:12.000 +0200	/Bilina/kamery/camea/BI-TP-I2	2025-10-26 15:04:33.922 +0100	-1	CZ	0	BI-TP-I2	1F4C42739F	2024-06-25 09:27:22.031 +0200	detector
7	127,537,485	2024-06-26 04:00:12.000 +0200	/Bilina/kamery/camea/BI-TP-I2B	2025-10-26 15:04:33.922 +0100	-1	CZ	0	BI-TP-I2B	92A8446D88	2024-06-25 09:27:22.378 +0200	detector
8	127,537,486	2024-06-26 04:00:12.000 +0200	/Bilina/kamery/camea/BI-MO-I1	2025-10-26 15:04:33.922 +0100	-1	CZ	0	BI-MO-I1	56C35F0585	2024-06-25 09:27:22.937 +0200	detector

Obrázek 3.7.: Staging tabulka v datovém skladu

3.3. Struktura a vrstvy datového skladu

Zdrojová data byla získávána ze systému Portabo, kde jsou ukládána jako MQTT zprávy v JSON formátu. Data mi byli poskytnuty zašifrované, abychom je mohli v rámci bakalářské práce využít. Tyto zprávy obsahují informace o detekcích vozidel – například registrační značku, typ detekce, rychlost, město, čas a senzor. Data z vodoměrů, elektroměrů a dalších senzorů. V rámci ETL procesu byla provedena tato fáze:

- **Data lake:** surová data načtená přímo z MariaDB SQLDump,
- **DW Staging:** očištěná, formátovaná a rozparsovaná data,
- **DW Faktová tabulka:** data transformovaná do hvězdicového modelu s vazbami na dimenze.

Každá implementace využívá dávkové zpracování (`BATCH_SIZE`) a kontrolu posledního zpracovaného záznamu (`LastLoadedID`), aby bylo možné ETL proces spouštět opakovaně a inkrementálně.

3.4. OLAP a vizualizace

V rámci praktické části byly nad vytvořenými datovými sklady vystavěny dvě samostatné analytické vrstvy – každá odpovídající jedné z testovaných technologií. Cílem bylo vytvořit funkční OLAP modely umožňující tvorbu interaktivních vizualizací, a současně porovnat rozdíly mezi komerčním řešením na platformě Microsoft SQL Server a open-source přístupem postaveným na nástroji Cube.js.

Komerční varianta – SSAS Tabular

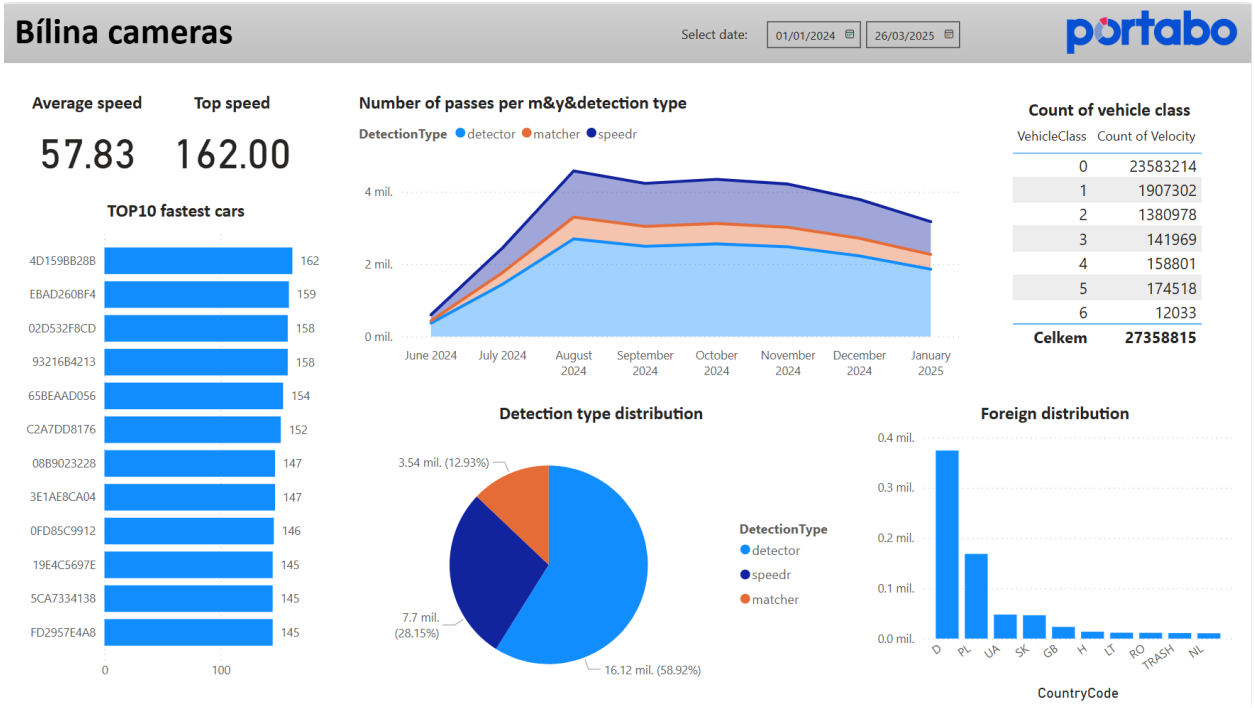
V komerční větvi datové platformy byl vytvořen datový model v prostředí **SQL Server Analysis Services (SSAS) Tabular**. Model byl vystavěn nad tabulkami `FactCameraDetection` a dimenzemi `DimCity`, `DimSensor`, `DimVehicleClass`, `DimDetectionType`, `DimLP` a `DimTime`. Propojení tabulek bylo definováno prostřednictvím relačních vazeb podle cizích klíčů, přičemž datový model respektuje hvězdicovou topologii (*star schema*).

Po načtení dat z datového skladu byla v prostředí SSAS vytvořena sada vypočítaných metrik v jazyce `DAX`. Jednalo se například o:

- `Detection Count` – celkový počet detekcí,
- `Average Velocity` – průměrná rychlost vozidel (vylučující nulové hodnoty),
- `Unique Plates` – počet unikátních registračních značek,
- `Detections Over Time` – časová agregace počtu detekcí podle dne a hodiny.

Součástí modelu byla také implementace základních hierarchií (například *Rok* → *Měsíc* → *Den*) a vybraných filtračních atributů pro přehlednější práci v Power BI. Po ověření funkčnosti byl

model nasazen na instanci `SSAS Tabular` a zpřístupněn uživatelům prostřednictvím **Microsoft Power BI** a **Excel PivotTables**. V Power BI byly vytvořen interaktivní report zobrazující přehled detekcí podle města, typu vozidla, senzoru a času. Některé metriky byly navíc doplněny o logiku pro kontrolu konzistence dat a filtrování extrémních hodnot. Výhodou tohoto přístupu byla plná integrace s ekosystémem Microsoft, vysoký výkon in-memory výpočtů a možnost implementace bezpečnostních pravidel na úrovni datového modelu.



Obrázek 3.8.: Ukázka výsledného PBI reportu

Open-source varianta – Cube.js a Apache Superset

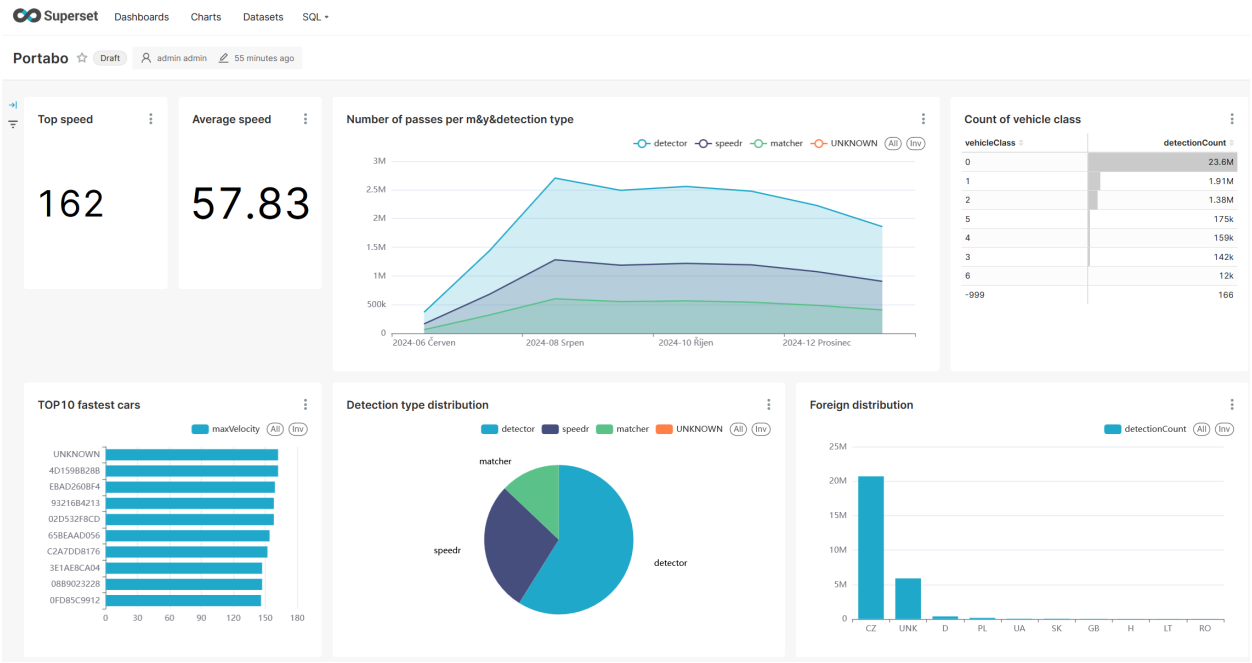
V open-source větvi datové platformy byla analytická vrstva implementována pomocí nástroje **Cube.js**, který byl nasazen nad databázemi **ClickHouse**, **MariaDB** a **PostgreSQL (TimescaleDB)**. Cube.js v tomto řešení plnil roli mezivrstvy typu OLAP, která sjednocovala přístup k datům z různých databázových systémů a poskytovala standardizované rozhraní pro vizualizaci.

Definice datového modelu byla realizována formou JavaScriptových souborů, přičemž hlavní model byl popsán v souboru `factcameradetection.js`. V tomto souboru byla definována analytická kostka `FactCameraDetection`, která obsahovala popis faktové tabulky a vazby na příslušné dimenze (`DimCity`, `DimSensor`, `DimVehicleClass`, `DimDetectionType`, `DimLP` a `DimTime`). Součástí definice byly také klíčové metriky odpovídající komerční variantě: celkový počet detekcí, průměrná rychlost, minimální a maximální rychlost, počet unikátních SPZ a počet detekcí s nenulovou rychlostí. Model využíval vazby typu `JOIN` na cizí klíče a odpovídal hvězdicovému schématu použitému v datovém skladu.

Po sestavení modelu byl **Cube.js** spuštěn jako samostatná `Node.js` služba, která poskytovala rozhraní typu **PostgreSQL API**. Toto rozhraní umožňuje klientským aplikacím komunikovat

s Cube.js stejným způsobem, jako by šlo o běžnou SQL databázi – tedy pomocí SQL dotazů. Díky tomu bylo možné na Cube.js napojit nástroj **Apache Superset**, který byl použit jako hlavní vizualizační platforma open-source řešení.

V rámci Superset byly vytvořen interaktivní dashboard zobrazující klíčové metriky: počty detekcí v čase, rozložení typů vozidel, přehled rychlostí. Data byla načítána prostřednictvím SQL dotazů směrovaných na PostgreSQL endpoint poskytovaný Cube.js, čímž byla zajištěna kompatibilita bez nutnosti dalších úprav. Pro zvýšení výkonu byly v Cube.js aktivovány funkce *pre-aggregations* a *query caching*, které umožňovaly ukládat výsledky často opakovaných dotazů.



Obrázek 3.9.: Ukázka výsledného Superset reportu

Porovnání přístupů

Oba přístupy poskytují plnohodnotnou OLAP analytiku, avšak liší se způsobem implementace i správy. **SSAS Tabular** umožňuje centrální řízení, pokročilé DAX výpočty a přímou integraci s Power BI, což zjednodušuje nasazení ve firemním prostředí. Naopak **Cube.js** nabízí flexibilitu, otevřenost a možnost úprav datového modelu přímo v kódu, což je výhodné zejména v agilním vývoji nebo při potřebě integrovat analytiku do webové aplikace.

Obě řešení tak umožnila plnohodnotnou vizualizaci a analýzu dat z detekčních systémů, přičemž volba konkrétní technologie závisí především na prostředí, infrastruktuře a požadavcích na rozšiřitelnost systému.

3.5. Provedení srovnávací analýzy

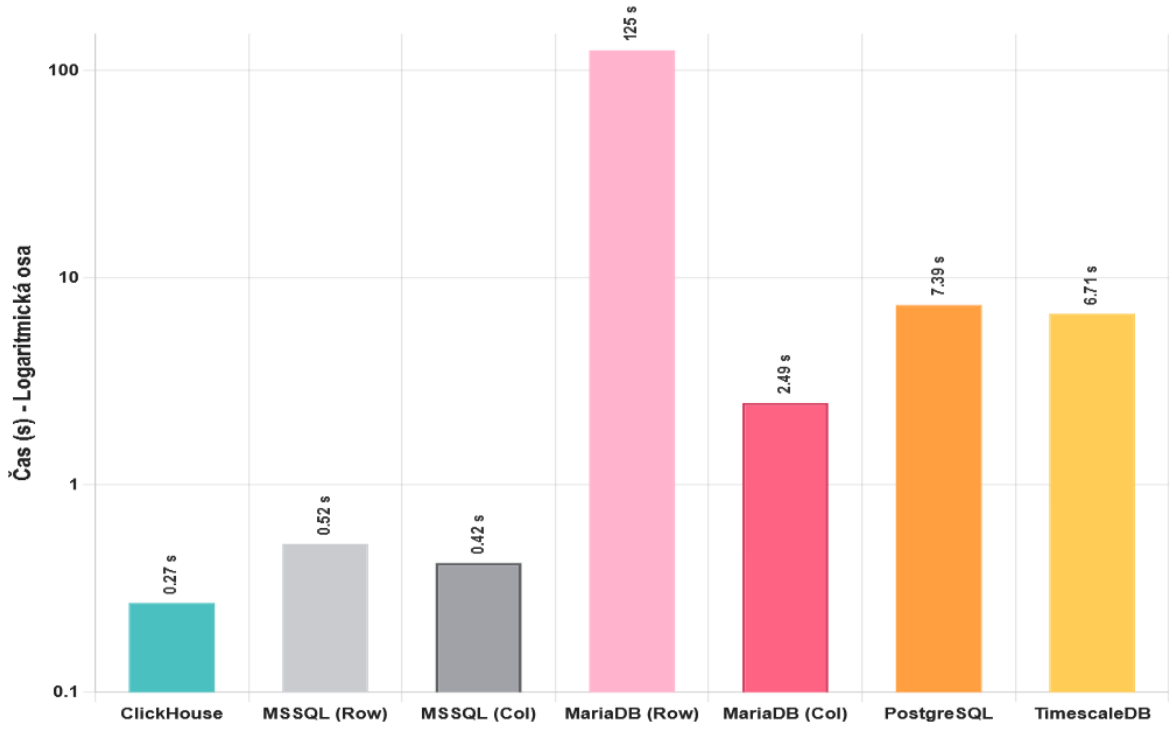
Cílem této kapitoly je objektivní srovnání výkonu a funkčních vlastností implementovaných datových architektur. Analýza byla záměrně koncipována jako **„out-of-the-box“ srovnání**, což znamená, že primárně hodnotí výkon a jednoduchost instalace jednotlivých systémů v jejich výchozí konfiguraci, s minimem dodatečných optimalizací. Tento přístup lépe odráží realitu organizací, které nemusí disponovat hlubokou IT expertizou pro pokročilé ladění.

Srovnání probíhalo ve dvou hlavních rovinách:

1. **Výkon na úrovni databáze:** Přímé měření doby odezvy SQL dotazů na různých databázových systémech a úložných enginech.
2. **Výkon na úrovni BI nástroje:** Hodnocení uživatelského prožitku, konkrétně rychlosti načtení a interakce s dashboardem v komerčním a open-source řešení.

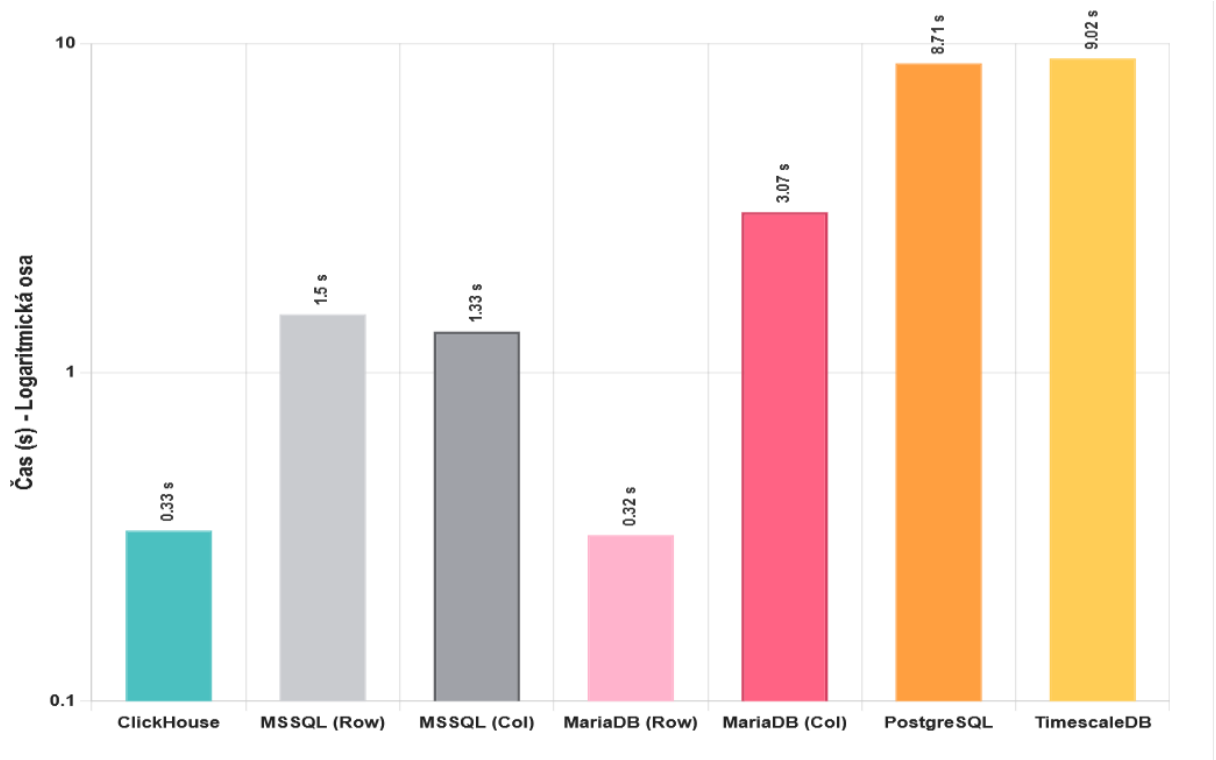
Srovnání výkonu na úrovni databáze

V první fázi byl měřen čas provedení sady pěti reprezentativních analytických dotazů přímo na databázových systémech. Cílem bylo porovnat efektivitu tradičního řádkového (row-store) a moderního sloupcového (column-store) úložiště napříč všemi testovanými platformami.



Obrázek 3.10.: Výkon jednoduché agregace s jedním spojením (Dotaz 1). Méně je lépe

Graf 3.10 ukazuje výsledky pro základní analytický dotaz, který počítá průměrnou rychlost pro každou třídu vozidla. Tento typ dotazu vyžaduje přečtení pouze dvou sloupců z hlavní faktové tabulky. Zde se naplno projevuje klíčová výhoda sloupcových databází (ClickHouse, MSSQL Col., MariaDB Col.), které jsou schopny načíst pouze data z potřebných sloupců. Naopak tradiční řádkové úložiště (MariaDB Row) musí načítat celé řádky, což při milionech záznamů vede k masivnímu I/O a extrémně pomalé odezvě.



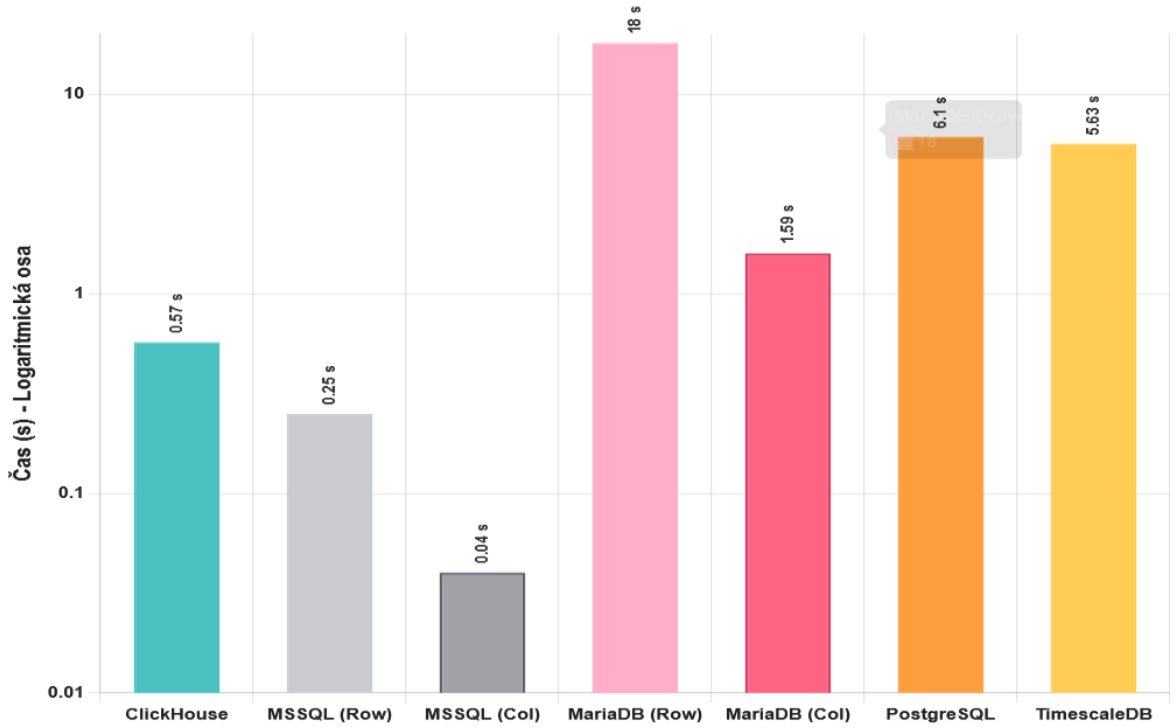
Obrázek 3.11.: Výkon agregace nad časovou dimenzí (Dotaz 2). Méně je lépe

Druhý dotaz sčítal počet detekcí pro každý den a odhalil zajímavé chování. Ačkoliv sloupcové databáze byly stále velmi rychlé, překvapivě dobrého výsledku dosáhla i řádková MariaDB. Tento jev je pravděpodobně způsoben efektivním využitím B-stromového indexu na sloupci ‘TimeKey‘ ve faktové tabulce. Řádkový engine dokázal díky indexu rychle provést spojení s dimenzí času a sečíst záznamy, čímž dočasně kompenzoval nevýhodu svého úložného formátu.

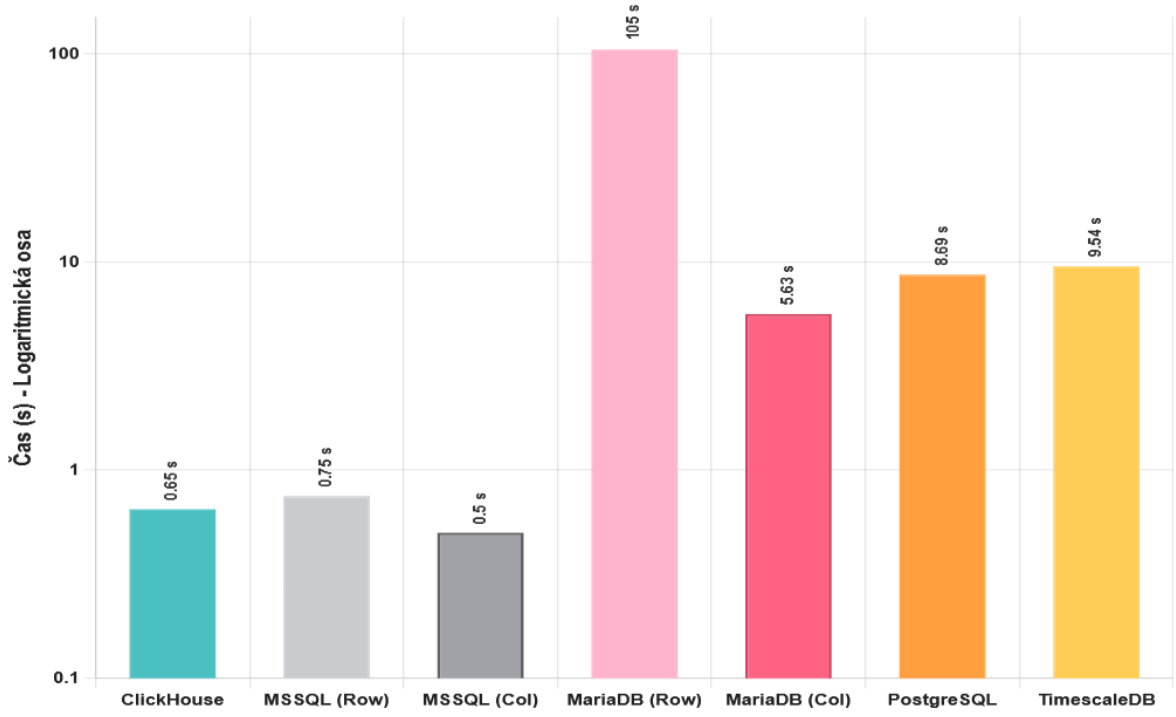
Třetí, komplexnější dotaz s několika ‘JOIN‘ operacemi a ‘WHERE‘ klauzulí, nejlépe demonstroval sílu moderních dotazovacích optimalizátorů. Extrémně rychlá odezva u MS SQL Serveru (Columnstore) a ClickHouse je dána jejich schopností aplikovat filtry co nejdříve v exekučním plánu (tzv. *predicate pushdown*). Tím se dramaticky sníží objem dat vstupujících do náročných operací spojení, což vede k řádovému zrychlení.

Čtvrtý dotaz, který agregoval data podle dvou dimenzí, opět potvrdil výhody sloupcového přístupu. Vyšší počet unikátních kombinací (kardinalita) při seskupování klade větší nároky na agregační engine. Výkonnostní propad řádkové MariaDB je zde opět fatální, zatímco všechny sloupcové varianty poskytují odezvu v řádu jednotek sekund, což je pro analytické účely přijatelné.

Poslední dotaz testoval schopnost databází zpracovat moderní SQL konstrukce jako Common

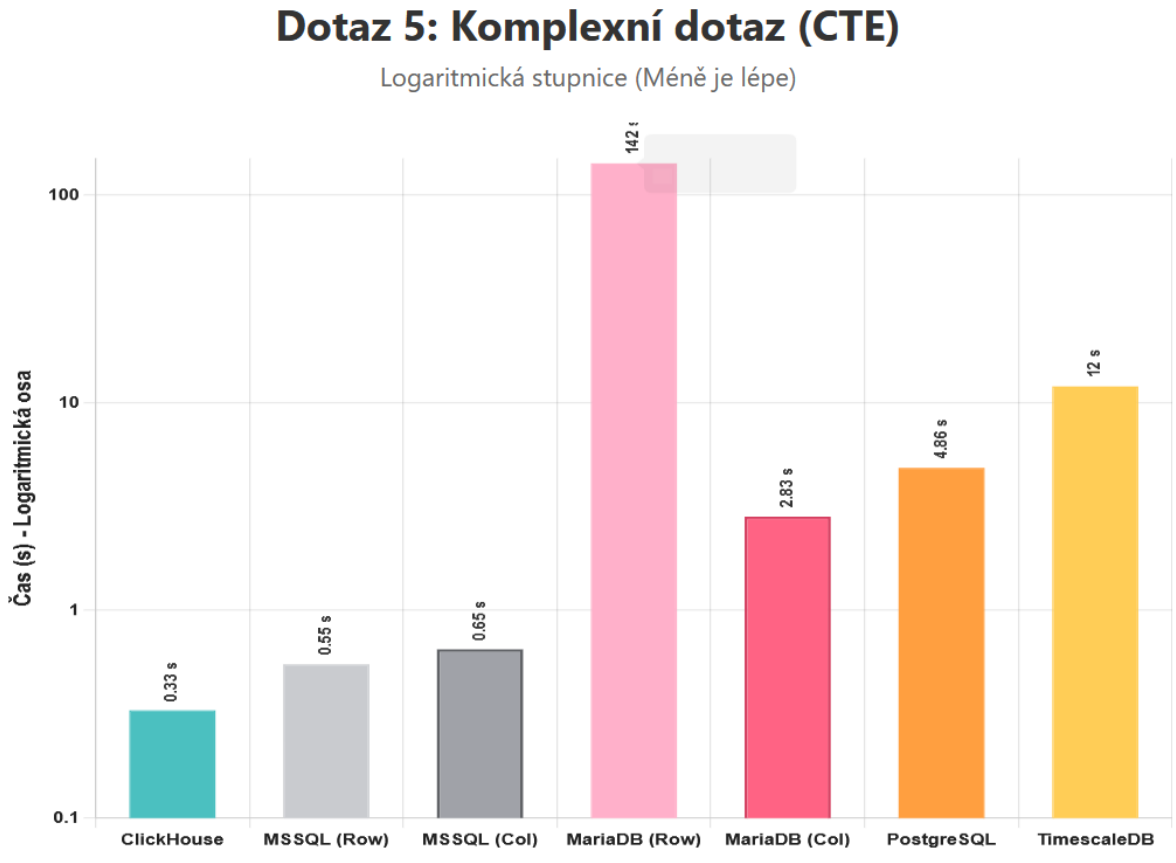


Obrázek 3.12.: Výkon dotazu s vícenásobným spojením a filtrováním (Dotaz 3). Méně je lépe



Obrázek 3.13.: Výkon agregace nad dvěma dimenzemi (Dotaz 4). Méně je lépe

Table Expressions (CTE) a okenní funkce. Graf 3.14 ukazuje, že databáze jako ClickHouse a MS SQL Server mají vysoce vyspělé optimalizátory, které si s touto komplexitou poradí efektivně. Naopak extrémně špatný výsledek řádkové MariaDB naznačuje, že její engine není pro tento typ analytických operací navržen a optimalizován.



Obrázek 3.14.: Výkon komplexního dotazu s CTE a okenní funkcí (Dotaz 5). Méně je lépe

Interpretace výsledků

Souhrnná analýza grafů demonstruje několik klíčových zjištění:

- **Nezbytnost sloupcového úložiště:** Nejvýraznějším poznatkem je propastný rozdíl ve výkonu mezi řádkovým a sloupcovým úložištěm pro analytické dotazy. U **MariaDB** se časy u komplexních dotazů zkrátily z více než 100 sekund na jednotky sekund. To jednoznačně potvrzuje, že pro seriózní BI je použití sloupcového enginu (nebo nativní sloupcové databáze) absolutní nutností.
- **Výkon nativních OLAP databází: ClickHouse**, jako databáze od základu navržená pro OLAP, dosahovala ve všech testech vynikajících výsledků a u většiny dotazů byla nejrychlejší. Její časy se konzistentně pohybovaly pod jednou sekundou.
- **Výkon komerčního řešení: MS SQL Server** s Columnstore indexem se výkonnostně velmi blížil ClickHouse a u dotazu č. 3 (spojení více tabulek s filtrem) byl dokonce nejrychlejší.
- **Role PostgreSQL:** PostgreSQL s rozšířením TimescaleDB, ačkoliv je optimalizován pro časové řady, v tomto obecném analytickém benchmarku zaostával za nativními sloupcovými řešeními. Jeho výkon byl sice výrazně lepší než u řádkové MariaDB, ale na ClickHouse či MS SQL Columnstore nestačil.

Srovnání výkonu na úrovni BI nástrojů a role pre-agregací

Druhá fáze testování se zaměřila na rychlost odezvy z pohledu koncového uživatele. Zde se ukázaly zásadní rozdíly plynoucí z odlišných architektur obou řešení.

Komerční varianta (Power BI + MS SSAS) funguje na principu **MOLAP (Multidimensional OLAP)**. Data z datového skladu jsou předem zpracována a kompletně načtena do tabulárního modelu v operační paměti. To sice vyžaduje delší čas při prvotním zpracování modelu, ale veškeré následné interakce uživatele v reportu (filtrování, proklikávání) jsou téměř okamžité (<1s), protože se počítají pouze nad daty v rychlé RAM. Pro koncového uživatele je tento prožitek velmi plynulý a komfortní.

Open-source varianta (Superset + Cube.js) naopak ve výchozím stavu pracuje na principu **ROLAP (Relational OLAP)**. Každý vizuální prvek na dashboardu generuje v reálném čase SQL dotaz, který je odeslán do databáze. Rychlost odezvy je tak přímo závislá na výkonu podkladové databáze. Jak ukázal test, při napojení na ClickHouse byly odezvy rychlé a srovnatelné s komerčním řešením.

Vysvětlení nekonzistentního výkonu pre-agregací

Během testování open-source varianty se podařilo zprovoznit **pre-agregace** v Cube.js, což vedlo k výraznému zrychlení. Byl však pozorován nekonzistentní výkon: někdy se dashboard obnovil za **2 sekundy**, jindy za **7 sekund**. Tento jev je klíčový pro pochopení fungování moderních BI akceleračních vrstev.

Příčinou je způsob, jakým Cube.js spravuje své pre-agregační tabulky (předpočítané, sumarizované tabulky):

- **Rychlá odezva (2s):** Nastává ve chvíli, kdy Cube.js obdrží dotaz a zjistí, že pro něj existuje již **postavená a platná pre-agregace**. V takovém případě se vůbec nedotazuje hlavní databáze (ClickHouse), ale vrátí výsledek přímo z této malé, optimalizované sumarizační tabulky. To je extrémně rychlé.
- **Pomalá odezva (7s):** Nastává tehdy, pokud pre-agregace ještě neexistuje, nebo ji Cube.js vyhodnotí jako **zastaralou** (neaktuální). To se děje na základě pravidla ‘refreshKey‘, které periodicky kontroluje, zda se zdrojová data nezměnila. Pokud ano, Cube.js musí pre-agregaci **znovu postavit**. Těchto 7 sekund tedy nepředstavuje čas dotazu uživatele, ale čas, který Cube.js potřeboval na pozadí ke spuštění dotazu proti velké faktové tabulce v ClickHouse, aby vytvořil novou, aktuální pre-agregační tabulku. Jakmile je jednou postavena, všechny další identické dotazy jsou opět rychlé (2s), dokud ‘refreshKey‘ opět nevyvolá její obnovu.

Shrnutí a interpretace

Souhrnná analýza potvrdila, že pro dosažení vysokého výkonu v interaktivní analytice je volba správné databázové technologie a architektury klíčová. Komerční řešení (Power BI + SSAS)

nabízí vynikající výkon a uživatelský prožitek „out-of-the-box“ díky své propracované in-memory (MOLAP) architektuře.

Open-source stack (Superset + Cube.js) dokáže nabídnout srovnatelný výkon, ale pouze za předpokladu, že je postaven nad vysoce výkonnou analytickou databází jako ClickHouse, nebo pokud jsou správně nakonfigurovány a spravovány **pre-agregace**. Právě zkušenost s pre-agregacemi ukázala, že ačkoliv jsou extrémně mocným nástrojem, jejich zprovoznění a pochopení jejich chování (cykly obnovování) vyžaduje hlubší technické znalosti. To potvrzuje závěr, že open-source řešení nabízejí obrovskou flexibilitu a škálovatelnost, avšak za cenu vyšších nároků na expertizu a čas věnovaný ladění.

Analýza celkových nákladů na vlastnictví (TCO)

Analýza celkových nákladů na vlastnictví (Total Cost of Ownership, TCO) je klíčovým faktorem při rozhodování o implementaci datové platformy. Zahrnuje nejen přímé náklady (licence, hardware, software), ale i nepřímé náklady (údržba, školení, expertíza, čas). V kontextu open-source a komerčních řešení se tyto náklady významně liší.

Open-source řešení (MariaDB, ClickHouse, Superset)

- **Nižší přímé náklady:** Open-source software je zpravidla zdarma, což eliminuje licenční poplatky. Náklady na hardware mohou být nižší díky efektivnějšímu využití zdrojů nebo možnosti provozu na méně výkonném hardware.
- **Vyšší nepřímé náklady:**
  - **Expertíza a vývoj:** Implementace a správa open-source řešení vyžaduje vyšší úroveň technické expertízy. Organizace musí investovat do školení vlastních zaměstnanců nebo najmout externí specialisty. Vývoj customizovaných řešení a integrace různých komponent může být časově náročný.
  - **Údržba a podpora:** Podpora je často komunitní, což může vést k delším dobám řešení problémů. Pro kritické systémy je nutné zvážit placenou podporu od třetích stran, což zvyšuje TCO.
  - **Časová náročnost:** Počáteční nastavení a konfigurace mohou trvat déle kvůli nutnosti integrace různých nástrojů a absenci jednotného ekosystému.
- **Flexibilita a vendor lock-in:** Vysoká flexibilita a minimální riziko vendor lock-in jsou významnými výhodami, které se však projevují v nepřímých nákladech na správu a integraci.

Komerční řešení (MSSQL, Power BI)

- **Vyšší přímé náklady:** Komerční software vyžaduje nákup licencí, které mohou být značně drahé, zejména pro velké podniky. Náklady na hardware mohou být také vyšší, pokud je vyžadován specifický hardware pro optimální výkon.

- **Nižší nepřímé náklady:**
  - **Integrovaná podpora a ekosystém:** Komerční řešení často nabízejí komplexní podporu od dodavatele, pravidelné aktualizace a dobře integrovaný ekosystém nástrojů. To snižuje potřebu interní expertízy a zkracuje dobu řešení problémů.
  - **Předvídatelnost a rychlost implementace:** Díky uceleným balíkům a profesionální podpoře je implementace často rychlejší a předvídatelnější.
  - **Školení a dokumentace:** Komerční produkty mají obvykle rozsáhlou dokumentaci a dostupné školicí programy, což usnadňuje zaškolení uživatelů.
- **Vendor lock-in:** Riziko vendor lock-in je vyšší, což může omezit budoucí flexibilitu a vyjednávací pozici s dodavatelem.

Závěr TCO analýzy

Volba mezi open-source a komerčním řešením závisí na strategii organizace, dostupných zdrojích a úrovni interní expertízy. Open-source řešení nabízejí úsporu na licenčních poplatcích, ale vyžadují značné investice do lidských zdrojů a času. Komerční řešení sice přinášejí vyšší počáteční náklady, ale mohou nabídnout rychlejší implementaci, nižší nároky na interní IT tým a vyšší předvídatelnost provozu díky komplexní podpoře. Pro platformu Portabo, která cílí na efektivní datovou analytiku bez nutnosti vysoké IT expertízy, by hybridní přístup kombinující silné stránky obou světů mohl představovat optimální řešení, kde se open-source komponenty využijí pro datový sklad a komerční pro vizualizaci.

3.6. Zhodnocení výsledků

Na základě provedené analýzy budou jednotlivé technologie porovnány z hlediska:

- **Technických parametrů** – výkon při dotazech, rychlost načítání dat, náročnost správy,
- **Uživatelské přívětivosti** – jednoduchost integrace s nástrojem Cube.js a možnost tvorby OLAP analýz,
- **Ekonomických aspektů** – licenční podmínky, náklady na provoz a škálování.

Závěrečná část shrne výhody a nevýhody jednotlivých řešení a poskytne doporučení vhodné platformy pro využití v rámci Portabo.

4. Sazba ukázek kódu

Sazba ukázek kódu je klíčová pro bakalářské práce věnované implementaci nějaké aplikace či knihovny.

Základní zásady pro sazbu ukázek kódu:

1.

5. Citace

Citace tvoří jeden ze základních pilířů závěrečné práce. Platí zde základní pravidlo: pokud použijete jakoukoliv zdroj informací, pak je nutné tento zdroj citovat, tj. uvést příslušný zdroj.

Zdrojem je ve většině případů text, ale může to být i obrázek, audiovizuální materiál či ve speciálních případech i ústní sdělení. V případě informatických prací je častým zdrojem u zdrojový kód.

Informaci ze zdroje můžete použít dvěma různými způsoby:

- přímo převzít (u textů je to známé Ctrl-C, Ctrl-V)
- použít jako základ vlastního intelektuálního výtvoru (textu, grafiky, programu, apod.), tj. použijete jen informaci, ale její formu změníte.

Prví druh tzv. přímé citace by měli mít v informatických závěrečných prací jen velmi omezený rozsah (méně než stránka), neboť jejich přínos pro hodnocení práce je diskutabilní. Přesto jsou však případy, kdy jsou vhodné:

1. matematické definice a tvrzení (věty, axiomy)
2. definice termínů z neinformatických oborů (např. společenských věd)
3. citace norem resp. standardů

Citace mají tři základní cíle:

1. určují, co je váš vlastní intelektuální přínos a co jste pouze převzali
2. pomáhají určovat primárního autora (resp. autory)
3. definují kontext vaší práce resp. mohou usnadňovat nalezení dalších souvisejících informací

Jakákoliv vědecká práce nevzniká na zelené louce a tak jsou citace její nezbytnou součástí. V rámci práce běžně navazujeme na existující výzkumy, projekty, technologie apod. Stejně tak se můžeme odkazovat na autority či s nimi polemizovat.

První dva cíle také úzce souvisejí s plagiátorstvím. Pokud v práci použijete myšlenku či údaj bez citování je vaše práce plagiátem. I když se to v obecném mínění vztahuje jen na přímé kopírování, není tomu tak. Přímé kopírování se jen snadněji vyhledává a prokazuje. Je také obtížnější jej kvantifikovat.

V případě přímého kopírování, jež není označeno jako přímá citace, postačuje i relativně malý rozsah (například věta, jeden obrázek, jedna procedura), aby byla práce označena jako plagiát.

Plagiát není možno obhájit a v případě většího rozsahu hrozí i vyloučení ze studia. Za přímé kopírování se považují i případy, kde je změna jen formální (změna slovosledu, náhrada synonym, zkrácení, vložení textové výplně, změna barvy či afinní transformace obrázku).

V případě převzetí myšlenky jde o zjevné plagiování, pokud je tato myšlenka důležitou částí práce (podílí se na splnění cílů).

To, že není plagiátorství odhaleno před obhajobou práce, není důkazem, že se nejedná o plagiát. Pokud je plagiátorství zjištěno později, může vám být odebrán titul i zpětně (a jak jste si jistě všimli, plagiátorství je běžně využíváno v politickém boji).

V každém případě si uvědomte, že plagiátorství je druh krádeže a že ani vy nechcete, aby někdo vaše myšlenky nebo dokonce váš text vydával za vlastní.

5.1. Označování citací

Označování citace má dvě části. Za prvé je nutno označit, jaká část práce je citací (rozsah) a jaký je původní zdroj.

Zdroj je vždy určen odkazem na bibliografický záznam, které jsou v případě bakalářské práce uvedeny v kapitole *Použité zdroje* na konci práce. Odkaz může mít různý tvar, ale preferovaný styl je uvedení čísla záznamu v hranatých závorkách. V případě použití našeho latexovského stylu stačí použít příkaz `\cite{id-zaznamu}`. V případě potřeby lze zdroj zpřesnit uvedením např. stránky či kapitoly, jež se uvádí za číslem záznamu (po čárce a ještě před uzavírající hranatou závorkou). V LaTeXu lze využít nepovinný parametr příkazu `cite`.

Označení rozsahu se poněkud liší u přímých a nepřímých citací.

U přímých citací je označení rozsahu kritické. V případě citací, které jsou kratší než odstavec je nutné text vyznačit kurzívou a zahrnout do uvozovek. Odkaz musí následovat hned za označným textem.

U citací v rozsahu odstavce či více odstavců se využívá zvětšení okrajů na levé i pravé straně odstavců (viditelné na první pohled). V LaTeXu lze použít prostředí `quote` nebo `quotation`. Text by měl být navíc v uvozovkách. Kurzíva je možná, ale u rozsáhlejších citací není příliš vhodná. Odkaz se umisťuje na konec posledního převzatého odstavce.

Speciální případ je možný v případě matematických definic a vět. Pokud jsou v rámci kapitoly převzaty jen z jediného zdroje, lze na počátku kapitoly uvést hromadný odkaz například v podobě věty: Všechny definice a věty uvedené v této kapitole jsou převzaty z [X].

V případě převzatých obrázků se odkaz umisťuje na konec popisku. Aby však bylo zřejmé, že se jedná o přímé převzetí (kurzívu ani uvozovky nelze použít) je nutné explicitně vyjádřit, že obrázek byl převzat ze zdroje bez podstatných změn například: (převzato z [X]) nebo (překresleno z [X]).

U nepřímých citací je vyznačování rozsahu volnější. Nejjednodušší je uvádění holé citace na konci vět (před tečkou) nebo konci odstavce (za poslední tečkou). V mnoha případech je ale možné citace uvádět explicitněji a stylistiky je provázet s okolním textem.

příklady:

- zajímavá alternativa je popsána v [x]
- údaj je je převzat z [x]
- použití návrhového vzoru poprvé popsal N.N v [x]
- volně přeloženo z [x]
- řešení bylo navrženo uživatelem N v [x] (vhodné např. pro stackoverflow a podobné zdroje)

Explicitnější vyjádření je nutno použít i v případě, že rozsah citace přesahuje odstavec.

- následující příklad je převzat z [x]
- výčet vychází z [x] je však doplněn o ...

Teoreticky lze podobné řešení využít i u celých sekcí či kapitol (kapitola je zpracována na základě [x]). V tomto případě je však nutné předpokládat, že v dané kapitole není žádná autorská myšlenka, a že autor se nesnažil najít alternativní pohledy či zdroje (a hodnotit tak, lze pouze autorovu schopnost výběru informací či stylistiky).

Výjimečně lze uvádět i několik citací se shodným či překrývajícím se rozsahem např. *následující specifikace je převzata z [x] a [y]*. To je však tolerovatelné jen v případě, v kdy by oddělení oddělení zdrojů bylo obtížné nebo nepřehledné a spojení nepřináší problémy s intelektuálním vlastnictvím (mají stejného autora či copyright). Zcela nepoužitelné jsou v případě většího rozsahu citace (např. na úrovni sekcí či kapitol)!

V případě obrázků je vhodné uvést explicitnější specifikaci, jak byl originální obrázek pozměněn resp, rozšířen.

Příklad:

- (převzato z [x] a doplněno)
- (převzato z [x], přeloženo)
- (upraveno z [x] pro novou verzi technologie ...)
- (inspirováno diagramem [x])
- (viz také [x] pro data X)

5.2. Bibliografický záznam

Biblografický záznam je datová struktura, jenž má dvě základní funkce:

1. jednoznačné identifikování zdroje
2. určení primární odpovědnosti (typicky je to autor resp. autoři, u webových zdrojů to však často bývá korporace).

Pro každý typ zdrojového dokumentu (zdroje) existuje množina klíčových atributů, které by měly být specifikovány (ne zcela vhodně označované jako povinné) a další, které hrají jen pomocnou roli.

V praxi však může nastat situace, kdy není zřejmé, jaký typ dokumentu pro daný zdroj zvolit resp. nelze zjistit hodnoty klíčových atributů. V tomto případě je nutné improvizovat a snažit se, aby záznam plnil v maximální míře obě funkce.

Struktura bibliografického záznamu je v zásadě dána těmito dimenzemi:

médium – základní dělení je na tištěné dokumenty a online dokumenty (dokumenty na elektronických nosičích tvoří jakási přechod mezi oběma typy dokumentů)

samostatnost – zdroj může být samostatný nebo součást rozsáhlejšího zdroje

periodičnost – periodický dokument vychází po jednotlivých částech, přičemž počet částí není předem znám (např. časopis).

Tištěné samostatné dokumenty neperiodické

Typickým příkladem samostatného tištěného dokumentu je kniha či monografie.

Základním zdrojem informací pro bibliografický záznam u knih je tzv. tiráž, tj. soupis vydavatelských údajů uvedený na konci knihy či na stránce za titulem. Využít lze i další zdroje (např. katalogy knihoven či knižní e-shopy, bibliografické záznamy v jiných dokumentech), ale v tomto případě je nutné provádět kontrolu, neboť tyto sekundární zdroje často obsahují chyby.

klíčové atributy:

ISBN : ISBN je celosvětový jedinečný identifikátor neperiodických tištěných dokumentů. Pokud ho kniha má, pak je dokument jednoznačně identifikován (a další identifikace už hraje jen sekundární roli). Pomlčky v ISBN nejsou součástí identifikátoru a lze je vynechávat (i když občas se jedno ISBN přiděluje více svazkům). Navíc existují ve dvou podobách ISBN-10 s deseti číslicemi a ISBN-13 s třinácti. Pokud jsou k dispozici oba je vhodnější uvídět ISBN-13 (i když ISBN-10 lze snadno mapovat na ISBN-13).

název : název knihy je povinný údaj a měl by být vždy vyplněn. Použit by měl být vždy originální název bez úprav. Jedinou přípustnou úpravou je změna velkých písmen (verzálek) na malá, které by mělo odpovídat pravidlům příslušného jazyka.

podnázev : některé knihy mají i podnázev Někdy je těžké rozeznat, co je název a podnázev. Zde platí pravidlo, že název by neměl obsahovat dvojtečku, tečku, středník apod. Od podnázvu je potřeba odlišit název edice. Podnázev je nepovinný (doporučuji uvádět pokud obsahuje klíčové informace).

autoři : v bibliografickém záznamu by měli být uvedeni všichni primární autoři (tj. není potřeba uvádět překladatele, ilustrátory, apod.)

vydání : označení konkrétního vydání. Je důležité především tehdy, když není známo ISBN a existuje více odlišných vydání (s různým obsahem)

nakladatel : uvádí se jméno nakladatelství, a to především z důvodů odpovědnosti

místo vydání : uvádí se jméno města, popřípadě stát, především tehdy pokud není jednoznačné (např. Cambridge) a to ve stručné podobě (např. stačí *United Kingdom*). Podobně stručný by měl být název nakladatelství (tj. bez označení typy společnosti, apod., rodičovské společnosti, apod.) V dnešní době globalizace je tento údaj v mnoha případech nevýznamný (tj. ho lze vynechat, především tehdy pokud je nakladatelství neznámé).

rok vydání : rok vydání přesněji identifikuje dokument. Pokud ho nelze zjistit, lze jej nahradit rokem copyrightu (v tomto případě je uvozen znakem `c` např. `c2022`)

edice : kniha může být vydána v rámci edice. Edici doporučuji neuvádět, výjimkou jsou edice, které jsou všeobecně známé.

URL : uvádí se pouze v případě, že je kniha dostupná onlině a to oficiálně a bez poplatků. Uvedení URL v tomto případě usnadňuje její získání (v tomto případě je ale často lepší citovat ji jako elektronickou knihu).

příklad:

Následující biblografický záznam byl získán z katalogu systému knihovny UJEP (volba Citace Pro v dolní části výpis záznamu).

RASCHKA, Sebastian a Vahid MIRJALILI. *Python machine learning: machine learning and deep learning with Python, scikit-learn, and TensorFlow.* Second edition. Birmingham: Packt, 2017. Expert insight. ISBN 978-1-78712-593-3.

Tento záznam splňuje základní požadavky, neboť obsahuje údaje týkající se odpovědnosti i jednoznačnou identifikaci dokumentu (a to jak ISBN tak přesným určením vydání). Zahrnutí podnázvu je vhodné, neboť obsahuje dodatečné informace (jména frameworků). Nakladatelství je uvedeno ve stručné podobě (tj. *Packt*) je uvedeno ve stručné podobě. Nadbytečné je jen uvedení edice (*Expert insight*).

Online samostatné dokumenty neperiodické

Typickým příkladem je online PDF dokument (včetně elektronické knihy). Dalším příkladem je webové sídlo (*web site*) tj. typicky hierarchický systém více stránek (nikoliv tedy jedna konkrétní web stran).

medium : u online zdrojů se jako médium uvádí slovo `online`.

URL : klíčový údaj pro online zdroje. Některé systémy (např. Wikipedia) poskytují tj. fixní URL, které odkazují na konkrétní verzi dokumentu, resp. stránek. I když jsou tato URL obecně delší, je nutné jim dát přednost, neboť zaručují jedinečnost.

název : název nelze vynechat i když ne vždy je jasné, co je hlavním názvem. V tomto případě je možné využít obsahu elementu title v hlavičce HTML (pokud je zdroj v HTML) nebo jiná metadata (například jak je zdroj pojmenován v odkazu).

autoři : autor nebývá u mnoha online dokumentů dohledatelný (a v tomto případě je nutné ho vynechat). Rozhodně však věnujte čas zjištění autorství (může být uvedeno i mimo dokument).

odpovědná korporace : typicky je to držitel intelektuálních práv (copyrightu). Důležitý je především v případě, že není znám autor, ale uvádějte ho ve všech případech, kdy je dohledatelný. Většina bibliografických stylů tento atribut nepodporuje resp. ho běžně nezobrazuje. Proto je vhodné pro tento účel využívat atribut *nakladatel* (i když to není totéž).

verze/čas poslední aktualizace : nahrazuje rok vydání. V případě, že není použit fixní odkaz, je klíčovým zdrojem informací, jaká z verzí dokumentu byla použita jako zdroj. Online dokumenty se mění často, a tak je vhodné uvádět, co nejpřesnější specifikaci (číslo verze, čas poslední aktualizace). Jen v případě, že dokument není verzován a nelze zjistit přesnější čas poslední modifikace, lze využít vročení (stejně jako u knih může být odhadnuto z copyrightu).

datum použití : je to povinný údaj i když důležitý je jen v případě, kdy nelze určit přesnější verzi. Měl by být v ISO formátu tj. ve tvaru RRRR-MM-DD. Toto datum běžně generuje editor bibliografických citací podle data vytvoření záznamu. V každém případě by mělo ležet v časovém intervalu od poslední modifikace zdroje (je-li uvedeno) do data odevzdání závěrečné práce.

příklad:

Dílčí tištěné dokumenty

U dílčích tištěných dokumentů je typické, že kromě identifikace dílčí části obsahují i identifikaci dokumentu jako celku.

Klasickým příkladem jsou články ve sborníku nebo vědeckém časopise. Kapitoly v knize (monografii) se citují, jen případě, že každou z nich vytvořil jiný autor (či kolektiv autorů)

V zásadě platí tato pravidla:

- uvádí se jen autoři dílčí časti, nikoliv například editoři sborníku nebo časopisu
- uvádí se pozice části v celém dokumentu nejlépe pomocí rozsahu stránek
- pokud má dílčí část vlastní jednoznačný identifikátor (například DOI), není potřeba uvádět identifikátor knihy nebo periodika.

Dílčí online dokumenty

Tento typ citací se používá pro webové stránky, jež jsou součástí webového sídla například pro konkrétní stránky s dokumentací nebo dokumenty uložené na GitHubu. Pro jiné elektronické dokumenty, pokud nejsou výslovně součástí webového sídla (např. elektronického sborníku) je vhodnější použít záznam samostatného dokumentu (viz výše).

Název stránky je doplněn jménem webového sídla (to je typicky uvedeno v záhlaví každé stránky resp. na hlavní stránce webového sídla. Autoři se vztahují ke stránce zatímco korporátní odpovědnost je typicky vztažena k celému sídlu (pokud jsou známy autoři i korporátní odpovědnost je vhodné uvést oba údaje, vždy však musí být uveden alespoň jeden z těchto údajů). Všechy ostatní atributy se vztahují

příklady:

What's New In Python 3.9: Summary – Release highlights. *Python 3.9.0 documentation [online]*. Python Software Foundation, October 14, 2020 [cit. 2020-10-15]. Dostupné z: https://docs.python.org/3/whatsnew/

Záznam obsahuje název dílčí části a také název celého webového sídla (v kurzívě). Odpovědná organizace je uvedena na místě nakladatele (autor není uveden a tak je tato informace klíčová). Verze je určena datem poslední modifikace (je uvedeno přímo ve tvaru použitém na stránce). Datum citování je povinné, ale v tomto případě nenese žádnou přidanou informaci (jen to, že citace byla vytvořena jen den po poslední modifikaci. Poslední součástí je URL.

Python nonlocal statement. *Stack Overflow* [online]. Stack Exchange, 2022-03 [cit. 2022-07-27]. Dostupné z: https://stackoverflow.com/questions/1261875/python-nonlocal-statement

Struktura záznamu je stejná. Čas poslední modifikace byl určen z informace, že poslední modifikace proběhla před čtyřmi měsíci (je uvedena v ISO formátu, ale odpovídající by byl i údaj například ve tvaru *březen 2022* nebo *March 2022*).

5.3. Často kladené otázky

Co není potřeba citovat?

Obecně platí, že citovat není potřeba znalosti, které jste získali v průběhu studia a to jak při výuce tak i z učebních materiálů (opor, skript). Citovat není potřeba ani zdroj formálních údajů (např. významu zkratek), pokud je lze snadno získat (například na Wikipedii).

To jest není nutné uvádět citaci při uvedení zkratky HTTP (zkratka je všeobecně známá a běžně využívána v mnoha kurzech). Podobně není nutné odkazovat pojmy jako Internet, počítačová síť, programovací jazyk, procesor, apod.

Běžně se také necitují (původní) myšlenky vedoucího práce, pokud si vedoucí práce nevyžádá jinak. Pokud vám zprostředkuje nepůvodní myšlenku, měl by vám pomoci najít originální zdroj (který uvedete v citaci).

Citování není možné v případě, kdy není znám původní zdroj, resp. je v podobě, kterou není možné citovat (lidová řčení, apod.) Pravděpodobnost výskytu takových textů v informatické bakalářské práci je však velmi nízká.

Jak citovat informace z (podnikových) školení

Pokud se jedná o evidentní výtvor školitele, můžete odkazovat příslušný výukový materiál (i když je neveřejný). Pokud je informace nepůvodní, pak je vhodné citovat primární resp. alespoň dostatečně autoritativní zdroj.

Jak citovat ústní sdělení?

Ústní sdělení je potřeba citovat jen tehdy, když je od autoritativní osoby v oblasti její odbornosti. Pokud například píšete práci o nasazení databáze, pak je autoritativní osobou například správce databázového systému (který vám sdělí například zkušenosti s nasazením).

Jak je uvedeno výše, ve většině případů se necitují ústní sdělení učitelů, školitelů, vedoucího práce a dalších sekundárních zdrojů.

Pokud citujete ústní sdělení je vhodné s tím danou osobu seznámit či získat alespoň neformální souhlas, neboť sdělené informace nemusí být veřejné.

Navzdory důležitosti ústních sdělení v některých typech prakticky zaměřených prací, není citace ústních sdělení standardizována. Jednoduchý návod nabízí například blog na citace.com [XXX].

Ústí sdělení je však neověřitelné a nelze ho jednoznačně identifikovat. Proto je lepší pokud se sdělení děje například e-mailem. Citace e-mailové komunikace i dalších netradičních zdrojů shrnuje dokument [XXX].

Je možno citovat Wikipedii?

Citování Wikipedie se obecně nedoporučuje, neboť se jedná o terciární zdroj (encyklopedia vytvořená na základě druhotných informací) a její kvalita je značně kolísavá.

Na druhou stranu Wikipedia (především v anglické verzi) často obsahuje i hodnotný a jinak jen obtížně dostupný materiál, a tak nelze citování z Wikipedie striktně zakázat.

Základní doporučení pro citování z Wikipedie:

- citujte jen tehdy, pokud nemáte k dispozici primární zdroje (ty jsou často odkazovány přímo z Wikipedie)
- citujte jen kvalitní články (které nejsou označeny jako problematické), které se v oblasti informatiky a matematiky objevují spíše na anglické Wkipedii
- citace z Wikipedie by měli tvořit jen malou část zdrojů (typicky méně než 10

Z Wikipedie rozhodně necitujte články věnované běžně známým technologiím a poznatkům, které jsou běžnou součástí kurzů.

6. Zhodnocení

7. Závěr

Závěr je klíčovou kapitolou, která může nejvíce ovlivnit vaši obhajobu. Základní částí závěru je přehledné shrnutí výstupů práce tj. co jste udělali pro dasažení cílů práce. Je nutné se vyhnout hodnocení, zda tím byli splněny cíle práce, či nikoliv (to je úkol posudků a především komise).

Seznam použitých zdrojů

1. BRAGIN, Tanya. *The Unbundling of the Cloud Data Warehouse*. ClickHouse Inc., 2023-11. Dostupné také z: https://clickhouse.com/blog/the-unbundling-of-the-cloud-data-warehouse. Accessed 2025-10-23.
2. PRIEBE, Torsten; NEUMAIER, Sebastian; MARKUS, Stefan. Von Data Warehouse bis Data Mesh: Ein Wegweiser durch den Dschungel analytischer Datenarchitekturen. *arXiv preprint arXiv:2212.03612*. 2022. Dostupné také z: https://arxiv.org/abs/2212.03612. Accessed 2025-10-23.
3. CUBE.DEV TEAM. *Semantic Layer and AI: The Future of Data Querying with Natural Language*. Cube.dev, 2024-08. Dostupné také z: https://cube.dev/blog/semantic-layer-and-ai-the-future-of-data-querying-with-natural-language. Accessed 2025-10-23.
4. SHARMA, Ayush. *Introduction to Cube.js – The Semantic Layer for Building Data Apps*. 2023-02. Dostupné také z: https://heyayush.com/post/2023-02-22_cube-js. Accessed 2025-10-23.
5. GEBRIM, Josue Luzardo. *Cube: Creating a Semantic Data Layer!* Medium, 2023-03. Dostupné také z: https://medium.com/codex/cube-creating-a-semantic-data-layer-a947fd0c11f6. Accessed 2025-10-23.
6. MICROSOFT CORPORATION. *Microsoft SQL Server Overview*. 2024. Dostupné také z: https://learn.microsoft.com/en-us/sql/sql-server/. Accessed 2025-10-23.
7. BUCK, Alex. *Data warehousing and analytics - Azure Architecture Center* [online]. [cit. 2025-11-14]. Dostupné z: https://learn.microsoft.com/en-us/azure/architecture/example-scenario/data/data-warehouse. Obrázek: Data warehousing and analytics architecture.
8. MICROSOFT CORPORATION. *Power BI Documentation*. 2024. Dostupné také z: https://learn.microsoft.com/en-us/power-bi/. Accessed 2025-10-23.
9. GOOGLE CLOUD. *BigQuery Documentation*. 2024. Dostupné také z: https://cloud.google.com/bigquery/docs/. Accessed 2025-10-23.
10. YADAV, Jitendra. *Implementing Data Products on Google Data Lakehouse Architecture* [online]. [cit. 2025-11-14]. Dostupné z: https://medium.com/google-cloud/implementing-data-products-on-google-data-lakehouse-architecture-df5c5f4cc07e. Publikováno na Google Cloud - Community (Medium).

11. AMAZON WEB SERVICES. *Amazon Redshift Overview*. 2024. Dostupné také z: https://aws.amazon.com/redshift/. Accessed 2025-10-23.

12. AMAZON WEB SERVICES. *Data warehouse system architecture* [online]. [cit. 2025-11-14]. Dostupné z: https://docs.aws.amazon.com/redshift/latest/dg/c_high_level_system_architecture.html. Obrázek: Amazon Redshift Architecture.

13. DBT LABS. *dbt Semantic Layer Documentation*. 2024. Dostupné také z: https://docs.getdbt.com/docs/use-dbt-semantic-layer/dbt-sl. Accessed 2025-10-23.

14. APACHE SOFTWARE FOUNDATION. *Apache Superset Documentation*. 2024. Dostupné také z: https://superset.apache.org/docs/. Accessed 2025-10-23.

15. METABASE INC. *Metabase Open Source Business Intelligence*. 2024. Dostupné také z: https://www.metabase.com/. Accessed 2025-10-23.

16. SERVICES, Amazon Web; TALEND. *Data Warehouse Modernization: From Monolith to Modular*. Amazon Web Services, 2019. Dostupné také z: https://pages.awscloud.com/rs/112-TZM-766/images/GLOBAL_PTNR_STPM_DWM_Talend_eBook_Aug-2019.pdf. Accessed 2025-10-23.

17. DATABRICKS. *The Modern Data Stack: How the Evolution of Data Architecture Led to the Data Intelligence Platform*. Databricks, 2024. Dostupné také z: https://www.databricks.com/blog/modern-data-stack-how-evolution-data-architecture-led-data-intelligence-platform. Accessed 2025-10-23.

18. WIKIMEDIA FOUNDATION. *Snowflake schema* [online]. [cit. 2025-11-07]. Dostupné z: https://en.wikipedia.org/wiki/Snowflake_schema. Použito pro ilustraci vločkového schématu. Obrázek: https://en.wikipedia.org/wiki/Snowflake_schema#/media/File:Snowflake-schema.png.

19. CLOUD, Google. *Looker Semantic Model — Unifying Business Logic for Data Experiences*. Google LLC, 2024. Dostupné také z: https://cloud.google.com/looker/docs/semantic-model-overview. Accessed 2025-10-23.

A. Externí přílohy

Externí přílohy této bakalářské práce jsou umístěny na adrese:
`https://github.com/Jiri-Fiser/thesis_ki_ujep`.

Na úložišti GitHub mohou byt uloženy tyto externí přílohy:

- **zdrojové kódy**
- **doplňkové texty** (například jak instalovat aplikaci, manuály aplikace)
- **schémata** (především, pokud se nevejdou na stranu A4 a jejich vytištění je tak problematické)
- **screenshoty** (v textu práce lze použít jen omezený počet snímků obrazovky, které navíc nemusí být při černobílém tisku příliš přehledné)
- **videa** (například ovládání aplikace)

V každém případě by to však měli být pouze materiály, které jste vytvořili sami. Materiály jiných autorů uvádějte v seznamu použité literatury (včetně případných odkazů na jejich originální umístění).

V této kapitole stačí uvést pouze základní strukturu úložiště (co se kde nalézá a jakou má funkci) například v podobě tabulky.

ki-thesis.pdf	text práce v PDF
ki-thesis.tex	zdrojový kód práce v LaTeXu
kitheses.cls	definice třídy dokumentů (rozšířená třída `scrbook`
thesis.bib	bibliografická databáze (exportována z citace.com)
LOGO_PRF_CZ_RGB_standard.jpg	logo fakulty s českým textem
LOGO_PRF_EM_RGB_standard.jpg	logo fakulty s anglickým textem

Všechny tyto soubory jsou potřeba pro překlad dokumentu (logo stačí jedno v příslušné jazykové verzi).

B. Další přílohy

Výjimečně může práce obsahovat i další tištěné přílohy. Obecně však dávejte přednost elektronickým přílohám umístěným na GitHubu (tato kapitola tak bude úplně chybět).