VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Roman Tomiczek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

RELAČNÍ A NOSQL DATABÁZE PRO SBĚR DAT Z IOT ZAŘÍZENÍ
RELATIONAL AND NOSQL DATABASE FOR COLLECTING DATA FROM IOT DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Roman Tomiczek

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Pavel Šeda

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Roman Tomiczek ***ID:*** 186215
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Relační a NoSQL databáze pro sběr dat z IoT zařízení

POKYNY PRO VYPRACOVÁNÍ:

Práce se bude zabývat analýzou využitelnosti relační a NoSQL (Not only SQL) databáze pro sběr velkého množství dat z IoT (Internet of Things) zařízení. Teoretická část práce se bude zabývat problematikou relačních a NoSQL databází, kde budou vybrány a popsány open source NoSQL databáze (MongoDB, Neo4j, ...) a relační databáze (MySQL, PostgreSQL, ...). Cílem práce bude návrh databáze pro ukládání dat z IoT zařízení monitorující pracovníka v kanceláři a implementace aplikace v jazyku Java, která umožní měření vybraných dotazů na databáze. Implementovaná aplikace bude vykreslovat výsledky z měření do přehledného grafu.

DOPORUČENÁ LITERATURA:

[1] FLEMING, Candace C. a Barbara. VON HALLE. Handbook of relational database design. Reading, Mass.: Addison-Wesley, c1989. ISBN 978-0201114348.

[2] HOLUBOVÁ, Irena, Jiří KOSEK, Karel MINAŘÍK a David NOVÁK. Big Data a NoSQL databáze. Praha: Grada, 2015. Profesionál. ISBN 978-80-247-5466-6.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 29.5.2018

Vedoucí práce: Ing. Pavel Šeda
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Bakalářská práce se zabývá analýzou relační a NoSQL (Not only SQL) databází pro sběr velkého množství dat z IoT (Internet of Things) zařízení. Teoretická část obsahuje seznámení se s relační a NoSQL databází. U relační databáze popisuje základní principy, relační algebru, normální formy atd. U nerelační databáze charakterizuje základní vlastnosti a popisuje různé typologie. Přiblíží také NoSQL dokumentovou databázi Elasticsearch. Praktická část obsahuje návrh a realizaci databáze MySQL, realizaci nerelační databáze Elasticsearch a JavaFx aplikaci pro měření parametrů databáze. Dále obsahuje výsledky měření parametrů a vlastností databází.

KLÍČOVÁ SLOVA

Databáze, MySQL, NoSQL, Elasticsearch, Java

ABSTRACT

This bachelor work is focused on the analysis of the relational and NoSQL (Not only SQL) databases for collecting a huge amount of data from IoT (Internet of Things) device. Theoretical part contains an introduction to the problematic of the relational and NoSQL databases. The relational database describes fundamental principles, relational algebra, normal forms and so on. The non-relational database characterises the main properties and describes different types of typologies. It deals with the NoSQL documentation databases called Elasticsearch as well. Practical part contains a proposal and realization of the MySQL database, realization non-relational database Elasticsearch as well as JavaFx application for measurement of database parameters. Further contains results of the measurement of parameters and properties of databases.

KEYWORDS

Database, MySQL, NoSQL, Elasticsearch, Java

TOMICZEK, Roman. *Relační a NoSQL databáze pro sběr dat z IoT zařízení*. Brno, 2018, 46 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Pavel Šeda

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Relační a NoSQL databáze pro sběr dat z IoT zařízení“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Pavlu Šedovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Většina aplikací musí nějakým způsobem uchovávat data. V současné době si lze jen těžko představit efektivní uložení dat bez použití databázových systémů. Mezi nejrozšířenější modely dat patří relační databáze. V současné době ale požadavky kladené na databáze rostou a klasické relační databázové systémy nejsou schopny těmto požadavkům vyhovět. Vznikl proto nový datový model nerelačních NoSQL databází.

V první kapitole jsou popsány principy a základní pojmy relační databáze. Relační databáze organizuje data do tabulek a všechny operace jsou prováděny na těchto tabulkách. Relace představují vztahy mezi tabulkami. Pro manipulaci s relacemi slouží relační algebra. Základem relační algebry jsou množinové operace (sjednocení, průnik, rozdíl a kartézský součin) a speciální operace (projekce, selekce a operace spojení). Dále jsou popsány normální formy, které slouží k efektivnější práci s daty a zabránění redundance dat.

Druhá kapitola je věnována NoSQL (Not only SQL) databázím. Je to nový druh databází, zaměřující se zejména na výkon. Hlavní výhodou nerelačních databází je při realizaci, kdy schéma spravovaných dat nemusí mít pevně danou strukturu, jako je tomu u relačních databází. NoSQL databáze lze rozčlenit na čtyři základní typologie. Databáze klíč–hodnota, dokumentové databáze, sloupcové databáze a grafové databáze.

V další kapitole je popsána NoSQL databáze Elasticsearch. Elasticsearch je dokumentová databáze, která obsahuje data v podobě dokumentů, psaných ve formátu JSON. Tato databáze umožňuje rychle ukládat a analyzovat velké objemy dat. Podporuje také fulltextové vyhledávání. K obsluze dat v Elasticsearch se používá grafické rozhraní Kibana.

Poslední kapitola se zabývá praktickou částí této práce. Je zde popsán návrh a realizace relační databáze MySQL, realizace nerelační dokumentové databáze Elasticsearch a popis aplikace v JavaFx pro měření rychlosti vykonávání příkazu na databázích. Dále jsou popsány měření, která byla provedena na databázích a která sloužila k porovnání těchto databází.

1 RELAČNÍ DATABÁZE

Relační databázové systémy jsou založeny na relačním modelu dat a relační algebře. Většina relačních databází používá SQL (Structured Query Language), což je jazyk, pomocí kterého se ovládá relační databáze. To zahrnuje vytvoření databáze, mazání, načítání řádků, úprava řádků, atd. SQL je ANSI (American National Standards Institute) standardní jazyk, ale existuje mnoho různých verzí jazyka SQL. Pro správu a údržbu dat se používá Database Managment systém (DBMS) což je software, který provádí veškeré operace s daty v databázi. Nejznámější DBMS jsou MySQL, MS Access, Oracle, SQL Server, PostgreSQL [1].

1.1 Relační algebra

Relační algebra se používá jako základ pro dotazovací jazyk SQL. Tuto skutečnost navrhl Doktor Edgar F. Codd. Relační algebra nepracuje s jednotlivými entitami ale s celými relacemi. Operátory relační algebry pracují pouze s relacemi a výsledky těchto operací jsou také relace. Mezi základní množinové operace patří sjednocení, průnik, rozdíl a kartézský součin. Relační algebra obsahuje i další operace, které jsou odvozeny od těch základních. Mezi ně patří selekce, projekce, přejmenování, spojení a rozdělení [2].

1.1.1 Množinové operace

Operátory relační algebry jsou matematické funkce používané k získání dotazů popisem sekvenčních operací v tabulkách nebo dokonce v databázích.
Sjednocení je označováno symbolem $\cup$. Cílem je sloučit všechna fakta z argumentů. Pro sjednocení musí být splněna následující podmínka. Domény n-tého atributu v jedné relaci se musí shodovat s n-tým atributem druhé relace Matematicky zapsáno jako: $A \cup B = \{x \mid x \in A \vee x \in B\}$
Průnik je označován symbolem $\cap$.Výsledkem je relace, která obsahuje všechny atributy počátečních relací. Matematicky zapsáno jako: $A \cap B = \{x \mid x \in A \wedge x \in B\}$
Sjednocení a průnik jsou operace komutativní a asociativní. To znamená, že platí následující vztahy: $A \cup B = B \cup A \;\; a \;\; A \cap B = B \cap A$
Rozdíl se označuje jako R1-R2. Výsledkem je relace, která obsahuje všechny prvky z první relace ale ne z druhé. Matematicky vyjádřeno jako:
$A - B = \{x \mid x \in A \wedge x \notin B\}$
Kartézský součin dvou relací je zapsán jako R1×R2. Před samostatnou operací součinu je nutné použít plně kvalifikované názvy atributů (připojit název relace před názvy atributů) [2].

1.1.2 Operace pro relační databáze

Operace pro relační databáze jsou operátory, které jsou odvozeny od základních operátorů.
Projekce je značená řeckým písmenem Π. Slouží k výběru sloupců a vytváří relaci nad podmnožinou atributů (sloupců).
Selekce pracuje podobně jako projekce, ale na rozdíl od ní vybere podmnožinu prvků (řádků). Značíme Σ.
Přejmenování slouží ke změně názvů atributů. Značí se řeckým písmenem ρ.
Operace spojení $\bowtie$ se používá pro seskupení prvků. Tato operace je důležitá pro databáze, které obsahují dvě a více tabulek, jelikož dokáži pracovat s více tabulkami naráz [2].

1.2 Coddova pravidla

Doktor Edgar F. Codd po rozsáhlém výzkumu relačního modelu databázových systémů přišel s vlastními dvanácti pravidly, která podle něj musí být splněna, aby byla databáze považována za skutečnou relační databázi. Tato pravidla mohou být aplikována na libovolný databázový systém, který spravuje uložená data pouze pomocí svých relačních možností.

- Informační pravidlo: Data uložená v databázi mohou být uživatelská data nebo metadata a musí být hodnotou některých buněk tabulky. Vše v databázi musí být uloženo ve formátu tabulky.
- Pravidlo zaručeného přístupu: Každý jednotlivý datový prvek je logicky přístupný kombinací názvu tabulky, primárního klíče a atributu.
- Systematické zacházení s hodnotami NULL: Hodnoty NULL v databázi musí být systematické a jednotné. Tyto hodnoty jsou podporovány pro vyjádření neznámé informace, a to nezávisle na datovém typu.
- Dynamický online katalog: Popis struktury celé databáze musí být uložen v online katalogu k němuž mohou přistupovat oprávnění uživatelé. Uživatelé mohou používat stejný jazyk dotazu pro přístup ke katalogu, který používají k přístupu k samotné databázi.
- Pravidlo komplexního datového jazyka: Databázi lze přistupovat pouze pomocí lineární syntaxe jazyka, která podporuje definici dat, manipulaci s daty a operace správy transakcí. Tento jazyk lze použít přímo nebo pomocí aplikace.
- Aktualizace pohledu: Všechny pohledy na databázi, které mohou být teoreticky aktualizovány, musí být také systémem aktualizovány.
- Schopnost vkládání, vytvoření a mazání: Schopnost zachování relačních pravidel u základních i odvozených relací je zachována nejen při pohledu na data,

ale i při operacích průniku, přidání a mazání dat.

- Nezávislost fyzických dat: Data uložená v databázi musí být nezávislá na aplikacích, které přistupují k databázi. Jakákoli změna ve fyzické struktuře databáze nesmí mít vliv na to, jak jsou data přístupná externím aplikacím.
- Nezávislost logických dat: Logická data v databázi musí být nezávislá na pohledu uživatele. Jakákoli změna v logických datech by neměla mít vliv na aplikace, které ji používají.
- Nezávislost integrity: Databáze musí být nezávislá na aplikaci, která ji používá. Všechna omezení integrity lze nezávisle měnit bez nutnosti jakékoli změny v aplikaci.
- Nezávislost distribuce: Konečný uživatel nesmí vidět, že jsou data distribuována na různých místech. Uživatelé by měli mít vždy dojem, že data se nacházejí pouze na jednom webu. Toto pravidlo bylo považováno za základ distribuovaných databázových systémů.
- Pravidlo nenarušení: Má-li systém rozhraní, které poskytuje přístup k záznamům na nižší úrovni, pak rozhraní nesmí být schopno obejít omezení zabezpečení a integrity [3].

1.3 Normalizace a denormalizace databáze

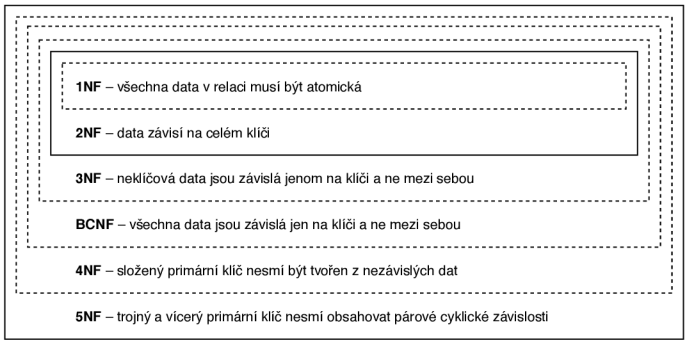
Normalizace je proces, kdy jednotlivá data jsou rozdělena do co nejmenších kompaktních celků, které jsou uloženy do tabulek, za účelem jednodušší práce s daty, zabránění redundance dat a lepší manipulaci s daty.

Normální formy jsou pravidla, podle kterých by se měly databáze navrhovat. Tím je myšleno návrh tabulek a relací mezi nimi. Celkem existuje šest normálních forem, kde čím vyšší normální forma, tím je lepší manipulace s daty. Každá vyšší normální forma obsahuje všechny pravidla nižších forem.

- 1NF – Všechna data v relaci musí být atomická. To znamená, že každé pole v tabulce musí představovat jedinečný typ informace.
- 2NF – Každý atribut, kromě primárního klíče, musí být plně závislý na primárním klíči. Tento problém se vyskytuje u tabulek, které obsahují dva a více primárních klíčů. Má-li tabulka jeden primární klíč, automaticky spadá do druhé normální formy.
- 3NF – Každý atribut, kromě primárního klíče, musí být závislý pouze na primárním klíči, a nesmí existovat závislost mezi ostatními atributy.
- Dále existují také Boyce-Coddova normální forma (BCNF), čtvrtá normální forma (4NF) a pátá normální forma (5NF). Ty se v praktickém návrhu databáze používají jen zřídka. Při absenci těchto pravidel u tvorby databáze ve

výsledku dostaneme méně dokonalý návrh databáze, ale funkčnost databáze to neovlivní [4].

Normalizované databáze mají data rozdělena do co nejmenších celků, což má své výhody, které jsou popsány výše ale také jednu podstatnou nevýhodu, a to snížení výkonu databáze. Normalizovaná databáze ke zpracování dat a dotazů využije více času, protože musí najít patřičné tabulky a sloučit jejich data pro dosažení požadovaných informací. Proto některé databáze pro zvýšení výkonu při dotazech jsou denormalizovány.



Obr. 1.1: Normální formy.

Denormalizace je postup, při kterém se databáze upraví tak, aby se zlepšil výkon databáze za příčinou řízené redundance dat. Pro upřesnění, denormalizovaná databáze není ta, která nikdy nebyla normalizovaná. Při denormalizaci databáze se úroveň normálních forem sníží o jeden či více stupňů. To způsobí, že klesne počet tabulek a zvýší se redundance dat. Dotazy na databázi budou rychlejší, protože většina dat bude víc pohromadě. Avšak budeme-li mít stejná data ve dvou tabulkách, při aktualizaci dat musíme obnovit data na dvou místech, jinak bychom měli v každé tabulce jiné údaje [5].

1.4 Transakce

U relačních databází se využívají 4 základní vlastnosti, které musí transakce splňovat. Jedná se o tzv. ACID. Složeno z anglických slov Atomicity, Consistency, Isolation, Durability.

- A – Atomicita (Atomicity): Transakce musí proběhnout vcelku. Vyskytne-li se nějaká chyba v průběhu transakce, pak je celá transakce zrušena.
- C – Konzistence (Cosistency): Transakce převede databázi z jednoho konzistentního stavu do jiného. Dojde k zápisu pouze platných dat, které dodržují integritní omezení.

- I – Nezávislost (Isolation): Je-li spuštěno víc transakcí v jednom okamžiku, tyto transakce se navzájem neovlivňují. To znamená, že provedením částečné změny v databázi v průběhu transakce se projeví až po dokončení transakce.
- D – Trvanlivost (Durability): Po dokončení úspěšné transakce jsou data uložena v databázi natrvalo [6].

1.5 Škálování

Existuje mnoho faktorů používaných ke zlepšení výkonu databáze. Jedním z nich je škálování. Škálování je schopnost systému, sítě nebo procesu zvládnout narůstající množství operací. Při škálování databází existují dva přístupy. Buď škálování vertikální nebo horizontální [19].

- Vertikální škálování – Označuje přidání dalších zdrojů v rámci stejné logické jednotky, pro zvýšení kapacity. Přidání CPU k existujícímu serveru, rozšíření úložiště přidáním pevného disku. Neboli zvyšování kapacity stávajícího hardwaru a softwaru. Databáze zde používá mnoho jader a procesorů, které sdílejí paměť RAM a disky.
- Horizontální škálování – Označuje koncept přidání více jednotek zdrojů a považuje je za jednu jednotku, jako jsou distribuované systémy. Je to tradiční model vyvažování zátěže a je to integrální součást v prostředí cloudu. Je to také schopnost distribuovat jak data, tak zatížení jednoduchých operací na mnoha serverech bez paměti RAM nebo disku, které by byly sdíleny mezi těmito servery. Horizontální škálování je považováno za modernější přístup.

1.6 Index

Hlavním účelem vytvoření indexu na konkrétní tabulce v databázi je rychlejší vyhledávání v tabulce a nalezení odpovídajícího řádku. Nevýhodou je, že indexy zpomalují přidávání řádků nebo aktualizaci stávajících řádků pro tuto tabulku a také obsazení úložného prostoru pro udržení indexové struktury dat. Přidání indexů může zvýšit výkon čtení a snížit výkon zápisu. Databázový index umožňuje dotazu efektivně načíst data z databáze. Indexy se vztahují ke konkrétním tabulkám a skládají se z jednoho nebo více klíčů. Tabulka může obsahovat více než jeden index. Klíče jsou fiktivní termín pro hodnoty, které chceme v indexu hledat. Klíče jsou založeny na sloupcích tabulek. Porovnáním klíčů s indexem je možné najít jednu nebo více databázových záznamů se stejnou hodnotou [15].

Vzhledem k tomu, že index drasticky zrychluje načítání dat, je nezbytné, aby byly pro každou tabulku definovány správné indexy. Chybějící indexy u malých databází

nebudou mít na výkon vliv, ale u větších databází můžou dotazy trvat delší dobu. Existují čtyři základní typy indexů:

- Index – Někdy označován jako sekundární. Je vytvářen ve sloupcích, kde se nenachází primární klíč. Slouží k efektivnímu hledání záznamů, které jsou jiné než primární klíč.
- Primární – Tento index se vytváří ve sloupci, ve kterém se nachází primární klíč. V tomto sloupci je každá hodnota unikátní a žádná hodnota není neznámá.
- Unikátní – Jedinečný index zajišťuje dostupnost pouze neopakujících se hodnot, a proto je každý řádek jedinečný.
- Fulltextový index – Podporuje efektivní vyhledávání slov v řetězcových datech [16].

1.7 Architekturu databázových systému

Architekturu databázových systému lze rozdělit na tři základní vrstvy. Jsou to vrstvy aplikační nebo uživatelská vrstva, logická a fyzická vrstva.

- Aplikační vrstva nebo vrstva uživatelského rozhraní představuje rozhraní pro všechny uživatelé. Poskytuje prostředky, pomocí kterých může uživatel komunikovat s databázovým serverem. Uživatelé můžeme rozdělit do čtyř základních skupin.
  - Sofistikovaní uživatele komunikují se systémem bez použití jakékoliv aplikace. Své požadavky tvoří s použitím dotazovacího jazyka.
  - Specializovaní uživatelé jsou aplikační programátoři, kteří píší specializované databázové aplikace.
  - Koncoví uživatelé, kteří se systémem vzájemně komunikují s použitím aplikace, která už byla předem vytvořena.
  - Správci databází, kteří mají úplnou kontrolu nad celým databázovým systémem. Mají široké spektrum odpovědností, včetně definice schématu, poskytování přístupu povolení atd.

  Všechny databázové systémy poskytují rozsáhlé služby pro všechny uživatelé z těchto skupin, které jsou ponechány na logické vrstvě, aby vhodně zpracovávaly různé požadavky [17].
- Logická vrstva obsahuje základní funkce řídicího systému relační databáze (RDBMS). Tato část systému obsahuje celou řadu implementací specifických pro dodavatele. Vztahuje se ke konkrétnímu databázovému modelu a používá jeho konstrukční dotazovací a manipulační prostředek.
- Fyzická nebo také datová vrstva je odpovědná za ukládání různých informací. Hlavní typy dat uložených v systému jsou:

– Datové soubory, které uchovávají uživatelská data.
– Indexy, které poskytují rychlý přístup k datovým položkám, které obsahují určité hodnoty.
– Statistické údaje, které uchovávají statistické informace o datech v databázi. Je používán dotazovacím procesorem, k výběru efektivních způsobů provedení dotazu.
– Logy, které se používají ke sledování provedených dotazů. Tyto data mohou být využita třeba při obnovení databáze v případě selhání systému.

1.8 Základní prvky

Základním prvkem relačních databází jsou tabulky, do kterých se ukládají data. Jednotlivé tabulky jsou propojeny předem nastavenými vztahy neboli relacemi. Tabulka je složena ze záhlaví, kde jsou uloženy názvy sloupců neboli atributů. Obsahuje také řádky (prvky). Průsečík řádku a sloupce se nazývá pole.

Pro komunikaci existuje strukturovaný jazyk dotazů (SQL), který je programovací jazyk používaný k návrhu relačních databází. V databázi SQL, jako jsou MySQL, Sybase, Oracle se provádí dotazy, načítání dat a úprava, aktualizace, mazání nebo vytváření nových záznamů. Jednou z konkrétních výhod SQL je její jednoduchá, přesto výkonná klauzule JOIN, která umožňuje vývojářům načíst související data uložená ve více tabulkách jediným příkazem.

1.9 Integrita

Termín integrita dat označuje přesnost a konzistenci dat. Při vytváření databází je třeba věnovat pozornost integritě dat a způsobu jejího zachování. Dobrá databáze bude vždy prosazovat integritu dat. Existuje pět druhů omezení integrity.

- Entitní omezení – Tato integrita zajišťuje, že každý záznam v tabulce je jedinečný a má primární klíč, který není NULL. To znamená, že v tabulce neexistuje duplicitní záznam nebo informace a každý záznam je jednoznačně identifikován nenulovým atributem v tabulce.
- Doménová integrita – Tato integrita zajišťuje, že jsou zde ověřeny jednotlivé sloupce tabulky, aby byly do sloupce zadány správné údaje. Například číselná data jsou zadána do sloupce NUMBER a nikoliv znak. Ve sloupci DATE jsou zadány správné datumy, a ne žádné neplatné hodnoty.
- Referenční integritní omezení – Tato integrita se věnuje vztahy mezi tabulkami. Proto, aby mohla být definovaná relace, musí být mezi příslušnými tabulkami

definován správný vztah pomocí primárních a cizích klíčů. Každý cizí klíč v tabulce by měl být primárním klíčem v související tabulce.

- Aktivní referenční omezení – Je aktivní v případě, že dojde k porušení integritních pravidel, a určuje co se v takovém případě bude dít.
- Sloupcové omezení – Toto omezení se liší od omezení domény, protože zde kontroluje platnost zadávaných dat. Zadává se správný věk, je zadáno správné ID zaměstnance apod. V doménovém omezení kontroluje, zda je zadána správná sada dat. Jako je datum zadáno do sloupce datum, číslo je zadáno do sloupce čísla. Toto omezení zaručuje, že hodnoty zadané do sloupce jsou správné. Například věk nebude záporný, nebo plat nebude záporný. Jedná se o obchodní pravidla nebo požadavky, které určují, jaký druh hodnot by mohl být zadán do každého sloupce [22].

1.10 Replikace

Replikace databáze je proces vytváření a údržba více instancí stejné databáze, procesu sdílení dat nebo změna návrhu databáze mezi databázemi v různých lokalitách, bez nutnosti kopírování celé databáze.

Synchronizace je proces zajištění, při kterém každá kopie databáze obsahuje stejné objekty a data. Při synchronizaci repliky se změněna provádí pouze u dat, která se změnila. Je možné také synchronizovat provedené změny v návrhu objektů. Zápisy databáze jsou odeslány do hlavního databázového master serveru a poté je návrh replikován do slave databázových serverů. Při čtení dat z databáze jsou požadavky rozděleny mezi všechny databázové servery, což vede k velké výkonnosti, díky sdílení zátěži. Kromě toho může replikace databáze také zlepšit dostupnost, protože slave servery mohou být nakonfigurovány jako master databázové servery pro případ, kdy hlavní databázový server nebude k dispozici [7].

Replikace, která představuje proces sdílení informací za účelem zajištění konzistence mezi redundantními zdroji, jako jsou softwarové nebo hardwarové komponenty, pro zlepšení spolehlivosti, odolnosti proti chybám nebo přístupnosti. Samotná replikace by měla být transparentní pro externího uživatele. V systémech, které replikují data, samotná replikace je aktivní nebo pasivní. Replikace dat, která se používá k vytvoření instancí stejných dat nebo části stejných údajů, se nesmí zaměňovat s procesem zálohování. Jelikož repliky jsou často aktualizovány a rychle ztrácí jakýkoliv historický stav. Mluvíme-li o aktivní replikaci, stejná žádost je zpracována v každé replikované instanci. V pasivní replikaci, je žádost zpracována na jednu repliku a pak je její stav převede na další repliky. Pokud existuje jedna hlavní replika určená ke zpracování všech požadavků, pak mluvíme o primárním schématu zálohování (režim master-slave) v clastrech s vysokou dostupností. Pokud jakákoliv replika zpracovává

požadavek a poté distribuuje její nový stav, pak je to multi-primární schéma, neboli multi-master.

2 NOSQL DATABÁZE

Nerelační databázový model označován jako No-SQL (ne SQL), později však spíše označován jako Not Only SQL (ne jenom SQL). Tento nový druh databází se zaměřuje hlavně na výkon. Výhodou u realizace nerelačních databází je, že schéma spravovaných dat se může měnit a nemusí mít pevně danou strukturu. U relačních databází taková možnost taky existuje, ale její provedení je poněkud složitější. Muselo by se upravit schéma databáze za běhu a tyto příkazy nemůže provést uživatel. Tyto operace jsou realizovány pomocí jazyka pro definici dat (DDL), které většinou provádí administrátor systému.

Tento typ databází je vhodný například pro ukládání velkých objemů dat, které často nemají žádnou strukturu. Databáze NoSQL neobsahuje žádné limity pro typy dat, které lze ukládat dohromady, a umožňuje přidávání různých nových typů podle potřeby. S databázemi založenými na dokumentech se může ukládat data, kde se typ dat nemusí předem definovat.

2.1 Transakce a CAP teorém

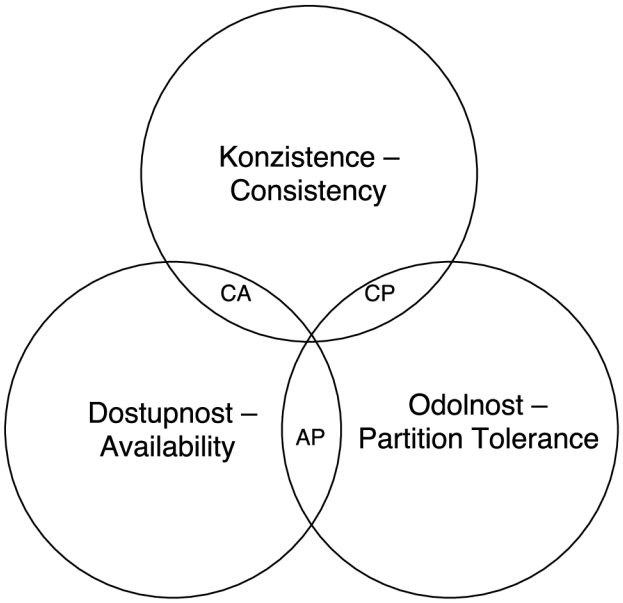
Na rozdíl od relačních databází, které používají transakce typu ACID, nerelační databáze využívají transakce typu BASE. Tento systém není tak striktní jako ACID, zato dovoluje dočasnou nekonzistenci dat. To způsobuje, že data při zápisu nejsou blokována, tím pádem je databáze výkonnější a rychlejší.

- Převážná dostupnost (basically available): Systém je převážně dostupný po celou dobu. K výpadku celého systému nedochází ale může dojít k částečným výpadkům.
- Volný stav (soft state): Systém je dynamický a nemusí být po celou dobu konzistentní.
- Občasná konzistence (eventual consistency): Někdy je systém uveden do konzistentního stavu, ale tu nemáme zaručenou neustále [8].

CAP teorém také známý jako Brewerův teorém. V nerelačních distribuovaných systémech bychom vlastnosti ACID vzhledem k distribuci, replikaci a částečným výpadkům systému nedodrželi. Proto používáme jiný přístup pomocí tří ideálních vlastností [6].

- Konzistence (consistenty): V určitém čase je k dispozici aktuální verze dat.
- Dostupnost (availability): Dostupnost znamená, že vždy dokážeme očekávat ukončení každé operace. Vysoká dostupnost se obvykle provádí prostřednictvím velkým počtem fyzických serverů fungujících jako jedna databáze. Prostřednictvím sdílení dat mezi různými databázovými uzly a replikacemi.

- Odolnost vůči rozpadu sítě (partition tolerance): Tolerance oddílů znamená, že pokud dojde k částečnému výpadku sítě, zbytek sítě musí nadále fungovat. Tolerance rozdělení lze dosáhnout mechanismy, které při zápisu dat určené pro nedosažitelné uzly odešlou data do uzlů, které jsou stále přístupné. Při následném obnovení sítě, uzly zpětně obdrží data [9].



Obr. 2.1: CAP teorém.

CAP teorém říká, že pro všechna sdílená data systému není možné zaručit současně všechny tři vlastnosti. Obzvláště ve webových aplikacích založených na horizontální strategii škálování je nutno rozhodnout mezi C a A. Obvyklé DBMS preferují C nad A a P. Existují dva směry při rozhodování, zda C nebo A. Jeden z nich vyžaduje silnou konzistenci jako základní vlastnosti a snaží se maximalizovat dostupnost. Výhoda silné konzistence, která připomíná transakce ACID, znamená pro vývoj aplikací datových služeb jednodušší správu. Na druhou stranu, komplexní musí být implementována logika aplikace, která detekuje a řeší nesoulad. Druhý směr upřednostňuje dostupnost a snaží se maximalizovat konzistenci. Priorita dostupnost má spíše ekonomické zdůvodnění. Nedostatečná dostupnost služby může znamenat finanční ztráty.

Databáze bez silné konzistence zapříčiní, že po zápisu data, ne každý kdo čte, uvidí správná data. Pokud opustíme silnou konzistenci, můžeme dosáhnout lepší dostupnosti, což zvýší škálovatelnost databáze. Takový přístup je vhodný pro cloudové databáze [9].

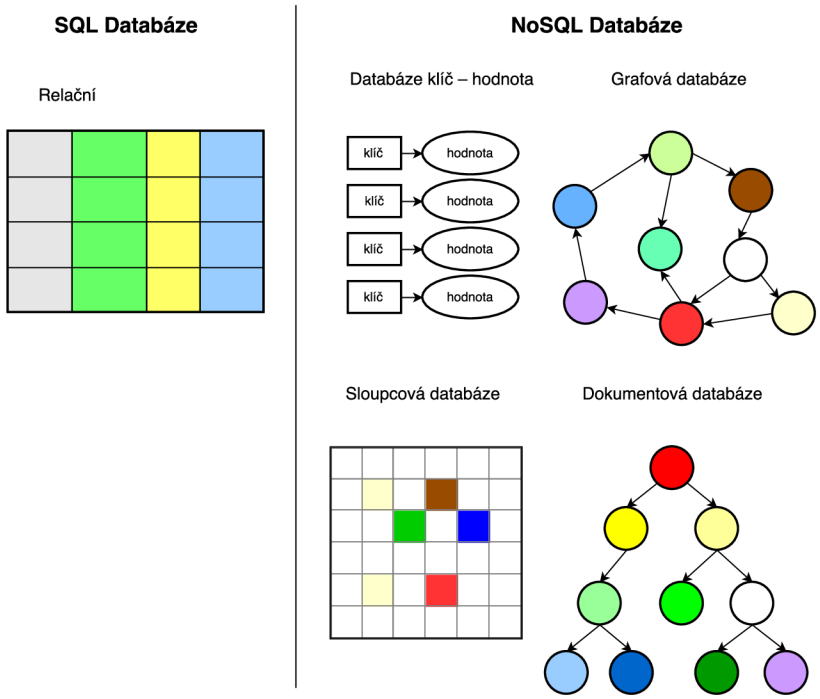
2.2 Typologie NOSQL

Datový model a databázové schéma lze navrhnout několika způsoby. U nerelačních databází lze tyto schémata rozdělit na čtyři základní typy. Databáze typu klíč – hodnota, dokumentové databáze, sloupcové databáze a grafové databáze.

- Databáze klíč – hodnota: Do těchto databází lze ukládat jakékoliv objekty, které se musí ukládat v páru společně s primárním klíčem. Tyto databáze jsou poměrně jednoduché, ačkoli neposkytují žádný způsob, jak s daty manipulovat.
- Dokumentové databáze: Systém bude obsahovat různé typy strukturovaných dokumentů. V dokumentové databázi je možné vyhledávat podle obsahu dokumentů.
- Sloupcové databáze: Tyto databáze ukládají data do tabulek, ale na rozdíl od tabulek v relačních databázích lze libovolně přidávat sloupce do řádků bez nutnosti je přidat do všech řádků.
- Grafové databáze: Tento druh databází je vhodný pro data, která je vhodné interpretovat jako grafy [8].

2.2.1 Srovnání a využití databází

Databáze typu klíč – hodnota – Jednoduchost databáze typu klíč-hodnota se ideálně hodí pro bleskově rychlé a vysoce kvalitní načítání hodnot potřebných pro aplikační úlohy, jako je správa uživatelských profilů, relací nebo získávání názvů produktů. To je důvod, proč Amazon ve svém nákupním košíku rozsáhle využívá vlastní systém Dynamo. Dynamo je vysoce dostupný systém ukládání klíčových hodnot, který některé z klíčových služeb společnosti Amazon používá k poskytování vysoce dostupného a škálovatelného distribuovaného úložiště dat.
Dokumentové databáze – Dokumentové databáze jsou vhodné pro ukládání a správu kolekcí velkých dat jako jsou textové dokumenty, e-mailové zprávy a dokumenty XML. Jsou také dobré pro ukládání semistrukturovaných dat, které by vyžadovaly rozsáhlé použití hodnot NULL v relačních databázích (hodnoty NULL jsou zástupnými symboly pro chybějící nebo neexistující hodnoty).
Sloupcové databáze – Sloupcový datový model je vhodný pro distribuované ukládání dat, zejména dat, které existují ve více verzích díky funkcím časového razítka. Dále jsou vhodné pro zpracování dat jako je třídění, konverze. Například konverze mezi hexadecimálními, binárními a desítkovými kódovými hodnotami.
Grafové databáze – Obecně platí, že grafové databáze jsou užitečné, pokud máte větší zájem o vztahy mezi daty než o samostatná data. Proto jsou grafové databáze optimalizovány pro získaní vztahů mezi daty a ne pro dotazování [10].



Obr. 2.2: Srovnání databází.

2.3 Integrované ukládání do mezipaměti

Řada produktů poskytuje mezipaměť pro SQL databázové systémy. Tyto systémy mohou podstatně zlepšit výkon čtení, ale nezlepšují výkon zápisu a zvyšují provozní složitost systému. Pokud aplikace převládá čtením z databáze, pak distribuovaná mezipaměť by mohla vylepšit vlastnosti databáze. Mnoho databázových technologií NoSQL má vynikající integrované možnosti ukládání do mezipaměti. Často používaná data uchovává v systémové paměti a odstraňuje potřebu ukládání do mezipaměti. Některé databáze systému NoSQL nabízejí plně řízenou a integrovanou vrstvu pro správu paměti v databází, které zajišťuje maximální výkon a nízkou latenci [21].

2.4 Dotazování databází NoSQL

Jazyk SQL používaný tradičními databázemi poskytuje jednotný způsob komunikace se serverem při ukládání a načítání dat. Syntaxe SQL je vysoce standardizovaná, takže zatímco jednotlivé databáze mohou některé operace zpracovávat odlišně, základy zůstávají stejné.

Naproti tomu každá databáze NoSQL má tendenci mít vlastní syntaxi pro dotazování a správu dat. Některé například používá žádosti ve formě JSON, odeslané

přes HTTP, pro vytváření nebo načítání dokumentů z databáze, jiné zase odesílají objekty JSON přes binární protokol prostřednictvím rozhraní příkazového řádku.

Některé produkty společnosti NoSQL mohou používat syntaxi typu SQL pro práci s daty, ale pouze v omezeném rozsahu. Například Apache Cassandra, sloupcová databáze, má svůj vlastní SQL jazyk, Cassandra Query Language nebo CQL. Některá syntaxe CQL je přímo převzata s SQL, jako jsou klíčová slova SELECT nebo INSERT. Neexistuje však způsob, jak provádět JOIN operace, jelikož zde neexistují relace, které jsou nutné pro tento typ dotazu, a tak tyto související klíčová slova v CQL neexistují [20].

2.5 Schopnost modernizace

Databáze, které podporují aplikace na webu, mobilní a internetové aplikace, musí být schopny pracovat v libovolném rozsahu. I když je možné měnit relační databázi jako Oracle, je to obvykle složité, drahé, a ne úplně spolehlivé.

Aplikace a služby musí podporovat neustále rostoucí počet uživatelů a dat. Neustále se musí rozšiřovat k udržení výkonu, a musí to dělat efektivně. Jedná se o problém relačních databází, které jsou omezeny vertikálním škálováním. Přidání více procesorů, paměti a úložiště na jediný fyzický server. V důsledku toho je schopnost efektivně rozšiřovat výkon docela výzva. Dále k provedení hardwarových upgradů může být databáze pozastavena, tudíž dojde k výpadku.

Distribuovaná databáze NoSQL však využívá horizontální škálování pro rozšiřování. Přidá více zdrojů jednoduše přidáním dalších serverů. Kromě toho, že je možné efektivně škálovat, distribuované databáze NoSQL se snadno instalují, konfigurují a rozšiřují. Byly navrženy tak, aby distribuovaly čtení, zápis a ukládání dat a aby fungovaly v jakémkoli měřítku, včetně správy a sledování malých i velkých klastrů [23].

2.6 Databáze typu klíč – hodnota

Do této databáze se data ukládají v páru. Kde první je klíč a druhá je hodnota. Hodnoty můžou být jakéhokoliv typu. Tyto databáze jsou velice rychlé a jednoduché. Operace, které poskytují manipulaci s daty jsou pouze tři.

- PUT – pro vložení hodnoty a klíče.
- GET – pro získání hodnoty podle klíče.
- DELETE – smazání páru klíč – hodnota.

Důležitou úlohu mají klíče, jako unikátní identifikátory, podle kterého se provádí veškeré operace v databázi. Velká část databází typu klíč – hodnota dovoluje

rozdělit data podle typu do takzvaných přihrádek. Přihrádkami můžeme rozumět jako jmenné prostory klíčů, které logicky izolují data stejného typu, ale fyzicky jsou všechny uloženy na stejném místě.

2.7 Dokumentové databáze

Hodnoty v těchto databázích budou uloženy do složených struktur neboli dokumentů. Dokumenty budou obsahovat samostatná data i metadata, které budou charakterizovat jednotlivé části dat. Základním formátem ukládaných dat je formát JSON, BSON (binární verze JSON) nebo XML. Dokumenty se nepoužívají jenom pro ukládáni dat ale také pro komunikaci s klientem. Přednost těchto databází je, že při ukládaní různých dokumentů, nemusíme měnit návrh databáze.
Pro manipulaci s daty neexistuje žádný standardizovaný jazyk. Prakticky každý systém, který používá dokumentovou databázi si vyvinul svůj dotazovací jazyk. Jedním ze systémů je MongoDB, jehož jazyk vychází z formátu JSON [8].

2.8 Sloupcové databáze

Základní idea sloupcově orientovaných databází je volnost přidávání sloupců a schopnost databáze účinně zvládnout tuto volnost. Výchozím prvkem je řádek označený klíčem řádku, který může mít libovolný počet sloupců. Každý sloupec obsahuje název sloupce, jeho hodnotu a časové razítko, kdy byla hodnota uložena. Jednotlivé sloupce jsou seskupeny do takzvaných rodin sloupců. Rodiny sloupců jsou důležitá část, jelikož při návrhu databáze definujeme pouze ji. Některé sloupcově orientované databáze umožňují vytvářet supersloupce. Hodnota supersloupce je pak složena z podsloupců [8].

2.9 Grafové databáze

Základními prvky grafových databázi jsou vrcholy a hrany. Vrcholy představují objekty a jejich hodnoty, a hrany představují vazby mezi objekty. Máme-li soubor objektů a jejich vztahy, můžeme podobnou datovou strukturu uložit i do relačních databází, které ale mají pár nevýhod. Jednou z nich je nízká efektivita při průchodu grafem, což je velké množství tabulek. Relační databáze data neukládají tak, aby bylo možné rychle a efektivně získat všechna data. Další nevýhodou je, že při návrhu relační databáze, musíme znát přesné schéma [8].

3 ELASTICSEARCH

Elasticsearch je nerelační dokumentová databáze. Data jsou uložena v podobě textových dokumentů psaných ve formátu JSON. Umožňuje rychle a v reálném čase ukládat, vyhledávat a analyzovat velké objemy dat.

3.1 Základní prvky

Dokument se skládá z polí (field), což je obdoba atributů v relačních databázích. Výhoda polí oproti atributů je, že se nemusí definovat předem ale pole se vytvoří automaticky na základě nahraného dokumentu. Pole může obsahovat maximálně jeden datový typ [11]. Architektura Elasticsearch je založena na následujících koncepcích:

- Index Elasticsearch je vytvořen a rozdělen na jeden nebo více částic, které se nacházejí na různých uzlech.
- Uzel je spuštěná instance modulu Elasticsearch. Když je uzel spuštěn, vyhledá cluster, který se k němu připojí.
- Cluster je skupina uzlů. Každý cluster je připojen k jednomu hlavnímu uzlu, který je vybrán automaticky.
- Hlavní uzel zpracovává primární shard, což je první místo, kde je dokument uložen, když je indexován. Po indexování dokumentu v primárním shardu se repliky primárního shardu také zkopírují.
- Replika shardu je jen kopie primárního shardu. Takže poskytuje záložní plán, jestliže primární shard spadne, a také replika shardu má schopnost zvýšit výkonnost Elasticsearch [12].

Index – Jednotlivé dokumenty jsou ukládané do indexu. Zde je možné definovat různé parametry uložiště pro všechny dokumenty. Jedním z parametrů je možné definovat typy (type), které označují dokumenty podobného formy. Typy můžeme přirovnat k tabulkám v relačních databázích. V praxi se však praktikuje ukládání různých dokumentů pod jednotlivé indexy.

Cluster – Elasticsearch je původně navržen tak aby fungoval v cloudu. Při vytvoření databáze se musí vytvořit cluster, který se umístí na více serverů, což umožní větší výkon databáze.

Shard – Pro distribuci indexů na více serverů se používají shardy. Ty označují fyzické rozdělení indexů. Hlavní výhoda rozdělení indexů je při vyhledávání, kdy hledání probíhá paralelně, což značně urychlí danou operaci. V prostředí cloudu, kde lze kdykoli očekávat výpadky, je velmi užitečné mít mechanismus převzetí služeb při selhání v případě, že dojde k výpadku serveru, na kterém je uložený shard. Za tímto účelem Elasticsearch umožňuje vytvořit jednu nebo více kopií shardu do takzvaných replik. Replikace je důležitá ze dvou hlavních důvodů:

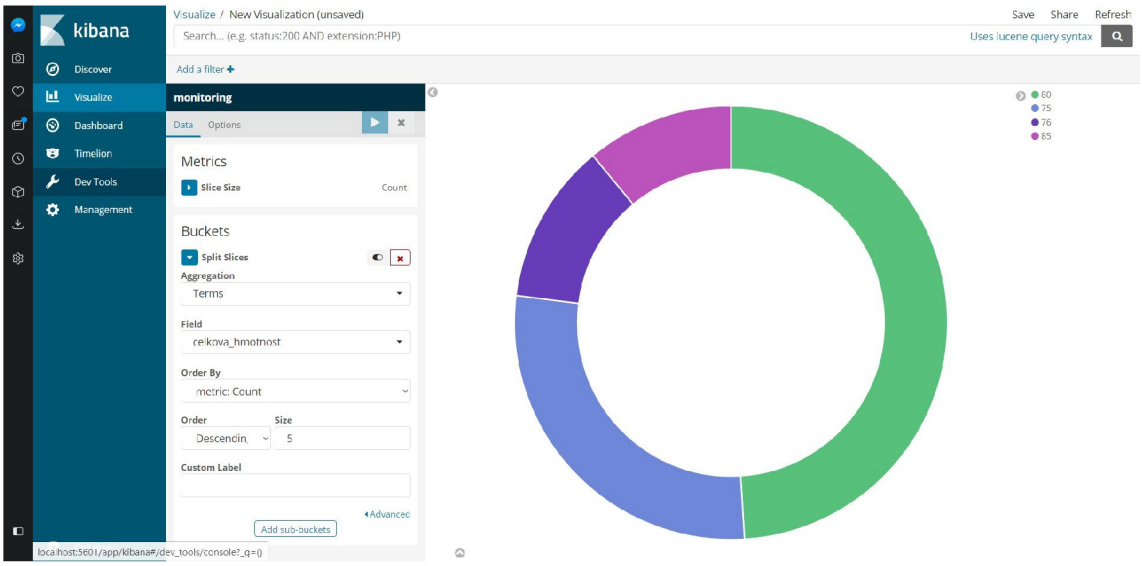
- Poskytuje vysokou dostupnost v případě selhání shardu nebo uzlu. Je však důležité, aby replika shardu nebyla uložena na stejném serveru jako originál.
- Umožňuje zvětšit propustnost vyhledávání, protože vyhledávání lze provádět na všech replikách paralelně.

Uzel – Uzel je samostatný server, který je součástí clusteru, ukládá data a účastní se indexování a vyhledávání v clusteru. Stejně jako cluster, uzel je identifikován názvem, který je ve výchozím nastavení náhodný, nastavený univerzálním jedinečným identifikátorem (UUID), který je při spuštění přiřazen uzlu. Později lze název uzlu definovat libovolným názvem. Tento název je důležitý pro správu, kde lze určit, které servery v síti odpovídají, jakým uzlům. Uzel lze nakonfigurovat tak, aby se připojil k určitému clusteru podle názvu clusteru. Ve výchozím nastavení je každý uzel nastaven tak, aby se připojil k clusteru s názvem elasticsearch. Jeden cluster můžete mít více uzlů. Pokud v síti nejsou žádné uzly Elasticsearch, při spuštění nového uzlu ve výchozím nastavení vytvoří nový cluster nazvaný elasticsearch [13].

Komunikace s elasticsearch je možná pomocí REST API. Některé dotazy na databázi je možné provést pouhým napsáním URL do webového prohlížeče. Vhodnější je pro práci s elasticsearch použít další nástroje. Jedním z nich ne Kibana, pomocí které je možné ukládání a vyhledávání dat.

3.2 Kibana

Kibana je grafické rozhraní, které slouží k obsluze dat uložených v elasticsearch. Základní využití kibany je vyhledávání dat a jejich následná vizualizace v podobě tabulek nebo grafů. Obsahuje také editor pro vykonávání příkazů. Hlavní výhody editoru je zvýraznění syntaxe psaného příkazu, automatické formátování dotazů, našeptávání při psaní dotazů a zobrazení historie psaných dotazů. Kibana usnadňuje pochopení velkých objemů dat. Jeho jednoduché rozhraní založené na prohlížeči umožňuje rychle vytvářet a sdílet dynamické panely, které zobrazují změny v dotazech Elasticsearch v reálném čase[13].



Obr. 3.1: Kibana.

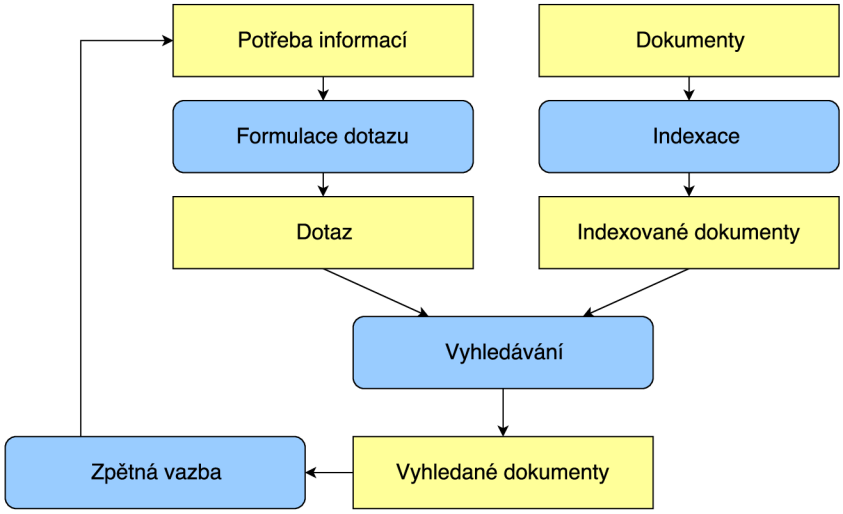
3.3 Transakce

Elasticsearch nepodporuje žádné transakce. Neexistuje žádný způsob, jak vrátit předložený dokument. V případě, nastane-li výpadek v průběhu operace, bude databáze v nekonzistentním stavu. Elasticsearch nabízí nastavení úrovně konzistence indexových operací. Lze nastavit počet replik, které musí potvrdit operaci před jejím návratem. Viditelnost změn je řízena při aktualizaci indexu, který je ve výchozím nastavení jednou za vteřinu a děje se na základě shard-by-shard. Elasticsearch se hlavně zaměřuje na rychlost a provádění distribuovaných transakcí zabere spoustu času [13].

3.4 Provedení vyhledávání

Elasticsearch poskytuje svůj vlastní dotazovací jazyk založený na JSON nazývaný Query DSL. Dané vyhledávání může být provedeno v Elasticsearch dvěma způsoby. Ve formě dotazu nebo formou filtru. Hlavní rozdíl mezi nimi je, že dotaz vypočítává a přiřazuje každý vrácený dokument s hodnocením významnosti, zatímco filtr ne. Z tohoto důvodu, vyhledávání přes filtry je rychlejší než prostřednictvím dotazů. Oficiální dokumentace doporučuje používat dotazy pouze ve dvou situacích. pro fulltextové vyhledávání nebo pokud relevance každého výsledku hledání je důležitá [14].

Fulltextové vyhledávání funguje na principu porovnávání hledaného slova nebo fráze s ostatními slovy v dokumentech. Pro efektivní a rychlé vyhledávání se vytvoří statistika výskytu slov a ta se uloží do databáze. Celá tato činnost se nazývá indexace.



Obr. 3.2: Fulltextové vyhledávání.

3.4.1 Indexace

Při indexaci jsou jednotlivé textové dokumenty uložené do speciálního indexu, kde jsou upraveny pro efektivnější vyhledávání. Úpravou je myšleno například použití jenom klíčových slov v dokumentu a transformováním je do základního tvaru. V elasticsearch se této transformaci říká analýza [13].

3.4.2 Analýza textu

Nastavení analýzerů je součástí konfigurace indexu. Při analýze se postupně provádějí následující operace:

- Filtrace znaků (character filters) – odstranění nechtěných znaků jako jsou html značky, interpunkce atd.
- Tokenizace (tokenizers) – rozdělení textu mezerami na slova neboli tokeny.
- Filtrace tokenů (token filters) – jednotlivá slova se upravují. Například převedení do prvního pádu, odstranění předpon nebo přípon, odstranění diakritiky nebo celých nepodstatných slov [13].

3.5 Mapování Implicitní a Explicitní

Mapování je podobné definici schématu v databázích SQL. Mapování je důležitou součástí každého indexu v Elasticsearch. Definuje, pro který index a typ je mapování vytvářeno. Dále jsou definované pole a jejich datové typy a popřípadě které pole jsou pro fulltextové vyhledávání. Elasticsearch může pracovat buď s implicitním nebo explicitním mapováním [11].

Implicitní mapování – Pokud server Elasticsearch nebyl mapován před vložením dokumentu, server se pokusí odvodit typ dokumentu, založený na hodnotách v polích dokumentu a přidá tento typ do mapování.

Explicitní mapování – Zatímco implicitní mapování může být adekvátním řešením, v některých případech je použití explicitního mapování výhodnější. Je pak možnost vytvořit složité typy dokumentů a kontrolovat, jak server Elasticsearch analyzuje každé pole. Explicitní mapování umožňuje zakázání indexování některých polí v dokument (výchozí server Elasticsearch indexuje všechny pole), což snižuje množství potřebného místa na disku a zvyšuje rychlost přidávání nových dokumentů. Tohle také poskytuje způsob ukládání dat, které nesmí být vyhledávány, ale musí být rychle přístupné prostřednictvím indexovaných polí [14].

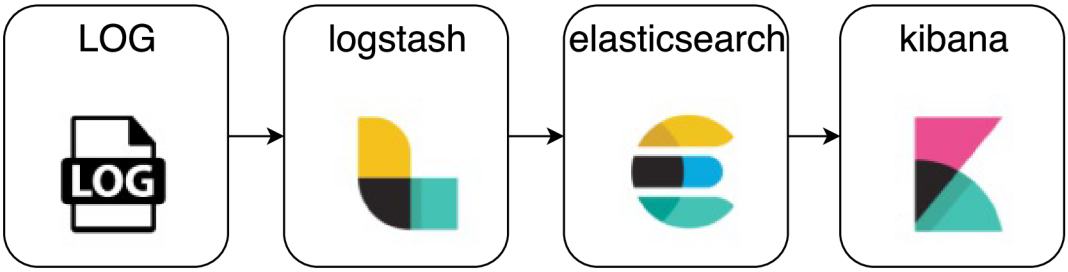
3.6 Log

Log slouží ke záznamu událostí, ke kterým dochází v systému, aby poskytl informace k pochopení činnosti systému a k diagnostice problémů. Jsou nezbytné pro pochopení činností složitých systémů, zejména v případě aplikací s malou interakcí uživatele (například serverových aplikací). Logy jsou rozděleny do šestí kategorií, které poslouží k určení závažnosti dané události.

- Chyba (Error) – Chyba znamená, že provedení některé úlohy nelze dokončit.
- Varování (Warn) – Varování znamená, že se stalo něco neočekávaného, ale tento proces může dále pokračovat. Varování jsou často znamením, že brzy se vyskytne chyba.
- Informace (Info) – Informace znamenají, že se stalo něco normálního, ale významného. Například systém byl spuštěn, systém byl zastaven, běžel denní inventář aktualizace úloh atd.
- Ladění (debug) – Tato zpráva znamená, že se stalo něco normálního a nevýznamného. Na webu přišel nový uživatel, stránka byla vykreslena, objednávka byla provedena, cena byla aktualizována. Tento typ zpráv je vyloučen z informací, protože by jich bylo příliš mnoho.
- Trasování (Trace) – Tento typ zpráv obsahuje velice podrobné informace a ve většině případů se ani nepoužívá [18].

Logstash je open source datový server, který pohlcuje data z mnoha zdrojů současně, transformuje je a poté je odešle do Elasticsearch. Data, která získává jsou z různých systémů, a tudíž i v mnoha formátech. Logstash podporuje celou řadu vstupů, které vybírají události z mnoha běžných zdrojů, a to současně. Jak data procházejí ze zdroje do úložiště, filtry Logstash analyzují každou událost a transformují je tak, aby konvergovaly na společný formát pro snadnější, urychlenou analýzu. Logstash má řadu výstupů, které umožňují směrovat data, kde chceme. Nejčastější výstup je Elasticsearch [13].

Elasticsearch následně slouží pro analýzu dat a pro fulltextové vyhledávání. Na konci řetězce je nástroj Kibana, která slouží pro správu dat a jejích vizuální zobrazení v grafech.



Obr. 3.3: Schéma logování.

3.7 Sledování aplikací v reálném čase

Elasticsearch nabízí možnost sledovat záznamy o aktivitách z aplikací a webových stránek. Elasticsearch indexuje data a zpřístupňuje je k analýze v téměř reálném čase, se zpožděním jenom jedné vteřiny. S využitím nástroje Kibana lze data vizuálně zobrazit v grafech nebo je použít k analýze a zjistit případné problémy. Elasticsearch také nabízí geoprostorovou analýzu, kterou lze využít k určení zeměpisné pozice, a určit, kde se případný problém nachází [24].

3.8 Bezpečnost

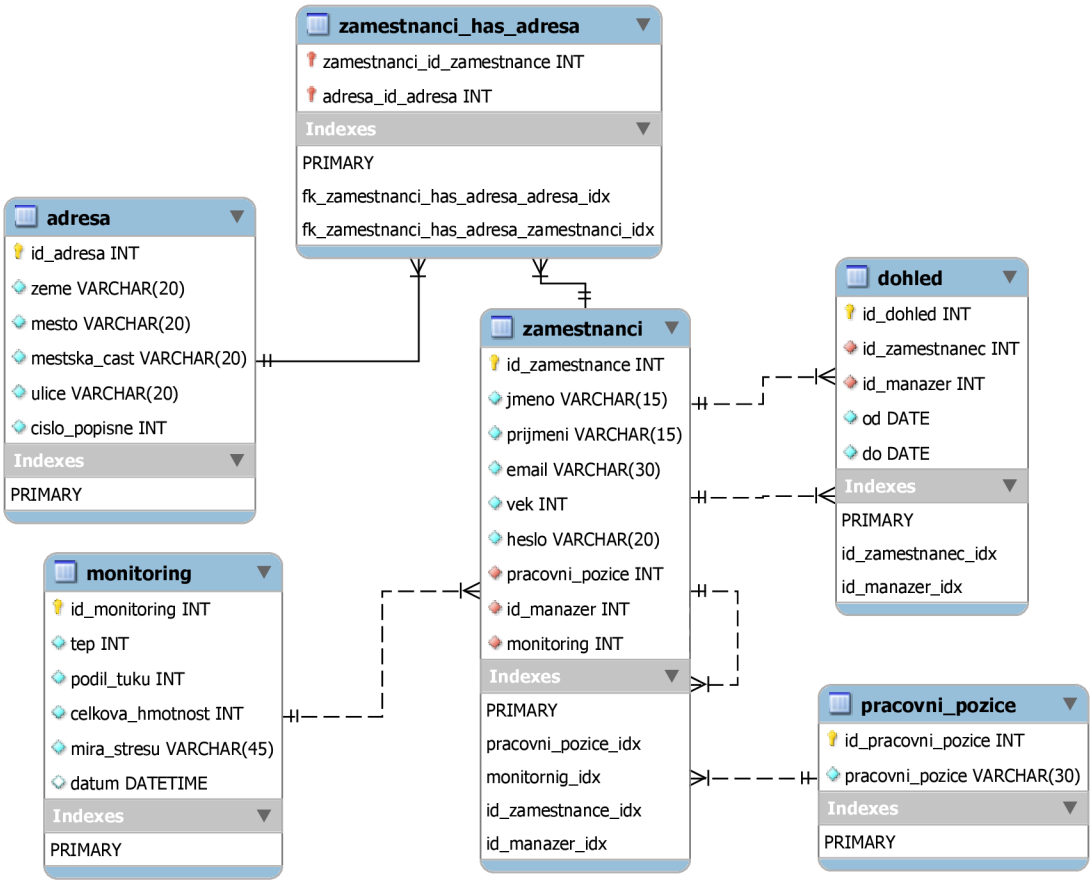
Z počátku, co se týkalo bezpečnosti, tak Elasticsearch neměl žádné funkce pro autentizaci ani autorizaci. Bylo tedy třeba zajistit bezpečnost na vlastní aplikační vrstvě a považovat Elasticsearch za systém, který zachází s každým uživatelem jako s důvěryhodným administrátorem. V dnešní době Elasticsearch podporuje produkt Shield, který nabízí komplexní zabezpečení [13].

4 PRAKTICKÁ ČÁST

Tato kapitola se zabývá návrhem a realizací relační databáze MySQL, dále obsahuje popis realizace databáze Elasticserch a popis aplikace JavaFx. V druhé části kapitoly jsou popsány měření, která byla provedena na těchto databázích.

4.1 Realizace MySQL databáze

Pro vytvoření relační databáze MySQL je nutné předem znát formát dat které se budou ukládat. Od toho se vyvíjí schéma databáze. V první řadě je nutné navrhnout tabulky a jejich atributy. Následně je třeba promyslet kardinality mezi tabulkami. Pro prvotní návrh databáze je použitý software MySQL Workbench.



Obr. 4.1: Vektorové schéma databáze.

Databáze obsahuje celkem šest tabulek.

- Hlavní tabulka *zamestnanci* obsahuje všechny základní údaje o zaměstnancích a také cizí klíče, které se odkazují na další tabulky.
- Tabulka *pracovni_pozice* obsahuje pracovní pozice zaměstnanců.
- Tabulka *monitoring* obsahuje všechny atributy monitorující zaměstnance.
- Tabulka *dohled* obsahuje id manažéra a id zaměstnance nad kterým má dohled. Dále obsahuje datum od–do.
- Tabulka *adresa* obsahuje všechny atributy týkající se adres. Na Obr.4.1 můžeme vidět i tabulku *zamestnanci_has_adresa*, která se vytvořila automaticky. Existence této tabulky naznačuje, že relace mezi tabulkami *zamestnanci* a *adresa* je M:N. To znamená, že jeden zaměstnanec může mít více adres a na jedné adrese může být více zaměstnanců.

Relace mezi tabulkami jsou následující. Tabulky *monitoring* a *pracovni_pozice* směřují k tabulce *zamestnanci*, která obsahuje cizí klíče. Naopak tabulka *dohled*, se odkazuje na tab. *zamestnanci*. Na tabulce *zamestnanci* můžeme vidět relaci, která se odkazuje na tu samou tabulku. Je to z toho důvodu, že každý zaměstnanec je dohlížen jedním manažérem. Kdyby byla vytvořena další tabulka manažerů, došlo by k duplicitě informací, jelikož každý manažér je i zaměstnanec. Všechny tyto relace jsou 1:N. Lze to vysvětlit na tabulkách *zamestnanci* a *pracovni_pozice*. Zaměstnanec může mít pouze jednu pracovní pozici, ale jedna pracovní pozice může mít více zaměstnanců.

Po vytvoření návrhu, se databáze nahraje na server phpMyAdmin. PhpMyAdmin je bezplatný softwarový nástroj napsaný v PHP. Jeho hlavním účelem ja správa MySQL databází přes web. Obsahuje grafické rozhraní, pomocí něhož lze vytvářet celé databáze, upravovat jednotlivé tabulky, definovat parametry atributů, spravovat uživatelské účty a oprávnění MySQL atd. Obsahuje také možnost exportu celé databáze do souboru a také import dat ze souboru. Data pro import můžou být až v sedmi formátech. Hodnoty, které byly importovány do této databáze, byly v souboru ve formátu .CSV.

4.2 Realizace databáze v Elasticsearch

Realizace nerelační databáze v Elasticsearch byla o něco jednodušší. Podle návrhu MySql databáze bylo vytvořeno pět dokumentů. Jelikož v Elasticsearch neexistují relace mezi dokumenty, nebylo potřeba vytvářet dokument z tabulky *zamestnanci_has_adresa*, která se vytvořila automaticky při návrhu relační databáze a naznačuje, že mezi tabulkami *zamestnanci* a *adresa* je relace typu M:N.

Následně vytvořené dokumenty bylo potřeba nahrát do databáze. Jelikož nerelační databáze jsou horizontálně škálovatelné, jednotlivý typ dokumentů se nahrává do jednoho indexu. Dokumenty se nahrávají pomocí nástroje Kibana, který obsahuje

editor pro vykonávání příkazů. Pro vytvoření nového dokumentu se používá příkaz POST. Dále se musí napsat název indexu a typ, které jednoznačně identifikují dokument. Pak už následují data, která jsou ve formátu JSON 4.2.

```
POST zamestnanci/zamestnanci
{
  "id_zamestnance": 1,
  "jmeno": "Melodee",
  "prijmeni": "Taber",
  "email": "okroeger@msn.com",
  "vek": 21,
  "heslo": 8542,
  "pracovni_pozice": 1,
  "id_manazer": "",
  "monitoring": 1,
  "adresa": 1
}
```

Obr. 4.2: Dokument ve formátu JSON.

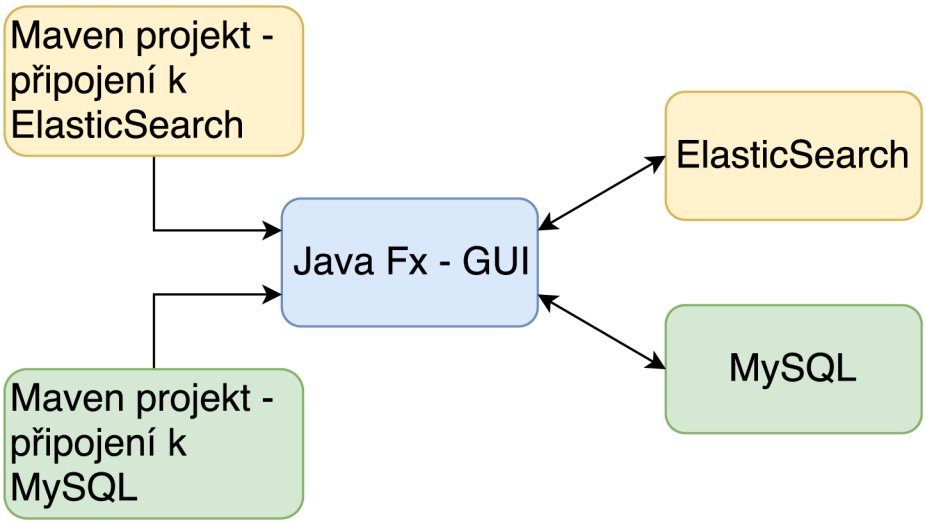
4.3 JavaFX aplikace

Aplikace je naprogramována ve vývojovém prostředí Eclipse. Při návrhu aplikace v JavaFX na obrázku 4.4, bylo použito tři projektů Maven v Java.

Projekt *elasticsearch_db_connector* slouží k připojení k Elasticsearch databázi. Tento projekt také obsahuje metody, které se používají k posílání dotazů do Elasticsearch. Java High Level Rest client jehož hlavním cílem je odhalit metody, které obsahují požadavky na databázi Elasticsearch a následně zpracovává tyto požadavky. Dále jsou zde metody, které definují jednotlivé příkazy vybrané v JavaFX aplikaci. Určují jejich podobu a parametry, například lze zde definovat maximální počet navrácených hodnot. Tyto metody se následně pošlou pomocí Java High Level Rest client.

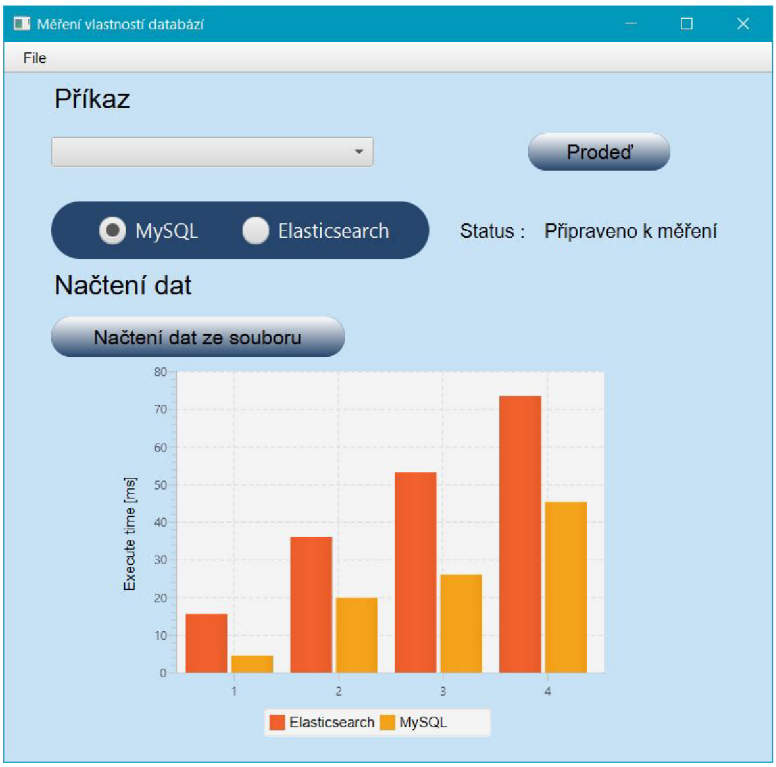
Další projekt *mysqlmysql_db_connector* slouží k připojení aplikace k serveru phpmyadmin, kde je uložená MySQL databáze. K přihlášení je nutné nastavit URL adresu databáze, login a heslo. Pro připojení k MySQL je použit singleton, neboli návrhový vzor. Singleton zabezpečuje, že v programu poběží pouze jedna instance této třídy.

Hlavní projekt *javaFx* sdružuje zbylé dva projekty, zároveň obsahuje grafické rozhraní aplikace. Aplikace nabízí možnost výběru k připojení buď k relační MySQL databázi, nebo k připojení k nerelační databázi Elasticsearch. Obsahuje taky metody, které měří rychlost vykonání požadavků na jednotlivé databáze.



Obr. 4.3: Schéma aplikace.

Po spuštění aplikace je třeba vybrat ke které databázi se má aplikace připojit. Vybere se jeden z předem definovaných příkazů, který se pošle do databáze. Aplikace nabízí možnost výběru ze čtyř příkazů. Každý z prvních tří příkazů se dotazuje na jednu tabulku nebo index. Poslední čtvrtý příkaz se dotazuje na dvě tabulky a dva indexy. Do relační databáze se posílá příkaz *select* s atributem *JOIN*, který slučuje tabulky a může odpovědět na příkaz, který požaduje data z více tabulek. Nerelační databáze ovšem neumožňuje slučovat data z různých indexů. Tento problém se vyřeší tak, že se pošle příkaz na oba indexy. Získaná data už ale musí zpracovat samostatná aplikace. Aplikace následně změří rychlost vykonání příkazu a výsledek uloží do souboru .CSV, který se uloží pod unikátním názvem podle data a času, takže výsledky v souborech jsou tak snadno dohledatelné. Aplikace také nabízí zobrazení výsledků měření ze souboru .CSV a zobrazí je ve sloupcovém grafu.



Obr. 4.4: Aplikace pro měření parametru databáze.

4.4 Měření

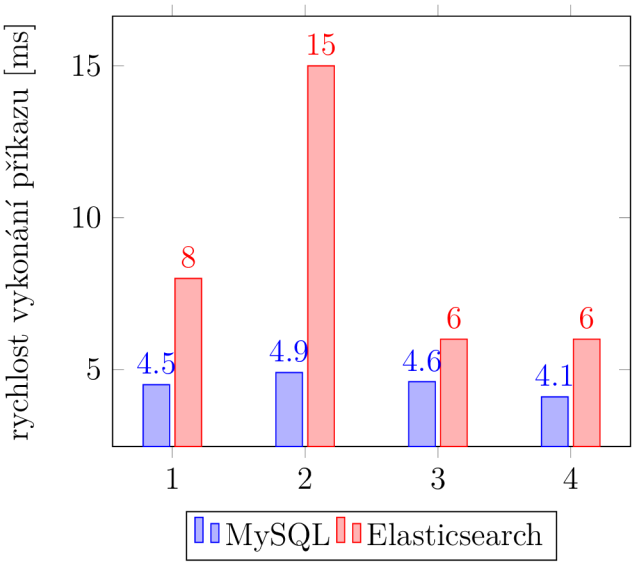
Pro srovnání databází byly provedeny tři typy měření, která porovnají MySQL a Elasticsearch v různých aspektech.

4.4.1 Rychlost vykonání příkazů

Pro změření rychlosti odezvy z databází je použita aplikace v JavaFX. Měření bylo provedeno na všechny čtyři dotazy, které aplikace nabízí. Data do databází byla nahrávána postupně, aby bylo možné zjistit, jaký vliv bude mít velikost vracených dotazů z jednotlivých databází na čase. Každý dotaz byl změřen pro 1000, 4000, 7000 a 10000 vrácených záznamů. Hodnoty rychlosti navrácení dotazu pro Elasticsearch byly změřeny z parametru TOOK, který měří nástroj Kibana a tento parametr uvádí rychlost vykonání příkazu v milisekundách, pouze na databázi Elasticsearch.

Z naměřených hodnot bylo zjištěno, že relační databáze MySQL pro dotazy, které byly pro 1000 záznamů, vykazovala rychlejší odezvy než nerelační databáze Elasticsearch. Rychlost dotazu se u MySQL pohyboval v rozmezí od 4,1 ms do 4,9 ms. U Elasticsearch se rychlost dotazu pohyboval od 6 ms do 15 ms. Největší rozdíl

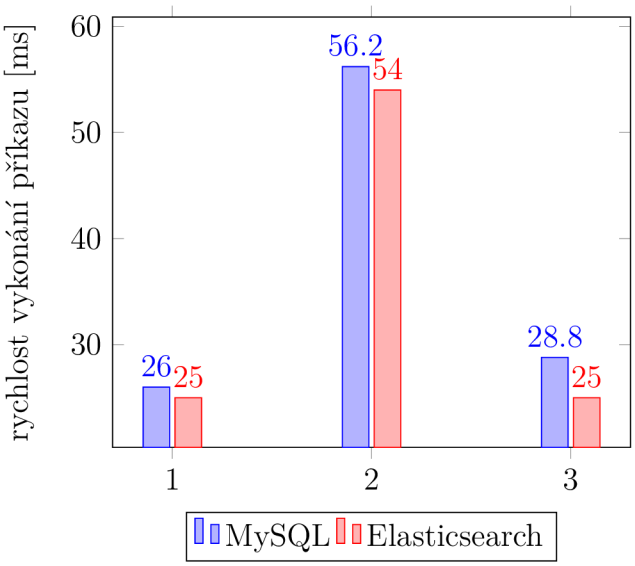
byl u dotazu na dokument se zaměstnanci, kde čas vráceného dotazu se lišil až o 10,1 ms.



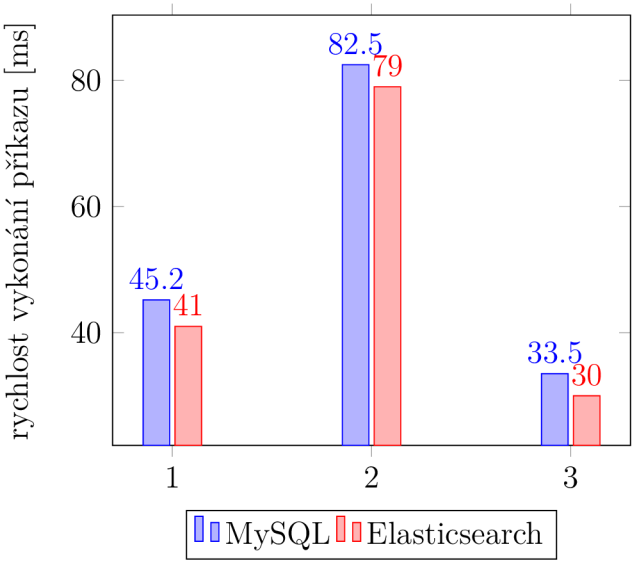
Obr. 4.5: Rychlost dotazů pro 1000 záznamů.

Dotazy, které obsahovaly 4000 záznamů a víc, byly rychlejší v nerelační databázi Elasticsearch. Největší rozdíl byl při dotazu pro 10000 záznamů, kdy výsledné hodnoty obou databází se lišily o 4 ms.

Obr. 4.6: Rychlost dotazů pro 4000 záznamů.

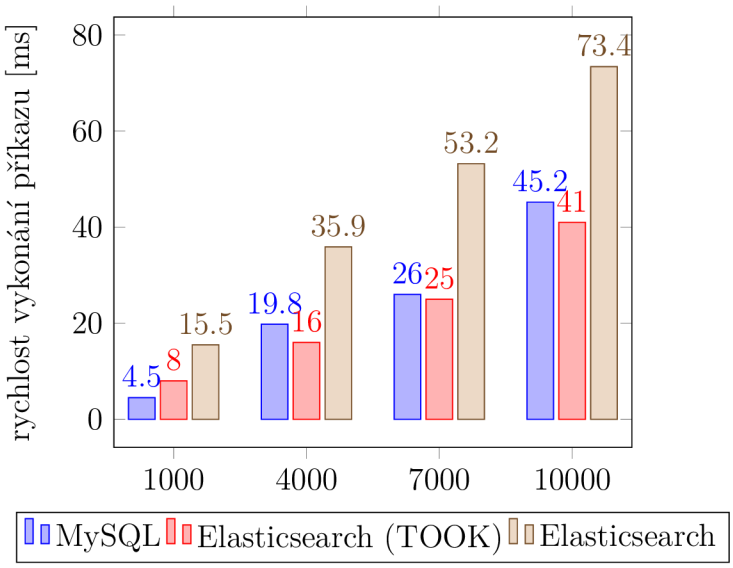


Obr. 4.7: Rychlost dotazů pro 7000 záznamů.



Obr. 4.8: Rychlost dotazů pro 10000 záznamů.

Aplikace v JavaFx měří čas vykonání příkazů na jednotlivých databázích i s přenosem dat. Zde byly hodnoty z databáze Elasticsearch výrazně pomalejší, než z MySQL.



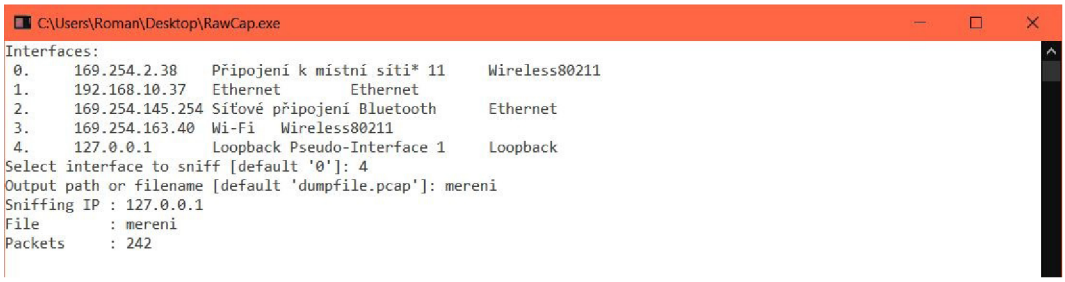
Obr. 4.9: Srovnání vykonání příkazů.

Na obrázku 4.9 je možné vidět, srovnání rychlosti vykonání příkazu pro dotaz na tabulku a dokument *monitoring* pro různý počet navrácených záznamů. Zde je vidět, že Elasticsearch výrazně zaostává za relační databází MySQL. Zpracování dotazu na samostatné databázi Elasticsearch je sice rychlejší ale i s přenosem dat je výsledná rychlost dotazu pomalá.

4.4.2 Síťová komunikace

Další srovnání je zatížení linky neboli kolik paketů a dat se posílá mezi jednotlivými databázemi při stejném dotazu. Jako testovací příkaz byl použit dotaz na tabulku a dokument *monitoring*, ze kterého se vrací hodnoty kde *celkova_hmotnost* se rovná 80. Maximální počet záznamů byl stanoven na 1000.

Pro zachycení komunikace byl použit program RawCap. RawCap na obrázku 4.10 je bezplatný software určený pro zachytáváni síťové komunikace na jakémkoliv rozhraní včetně ip adresy 127.0.0.1, což je lokální smyčka, na které probíhá veškerá komunikace mezi aplikací JavaFX a databázemi. Výsledná zachycená komunikace se uloží do souboru. Pro zobrazení a následnou analýzu dat byl využit program Wireshark, který je jedním z nejpoužívanějších analyzátoru síťové komunikace.



Obr. 4.10: Uživatelské rozhraní programu RawCap.

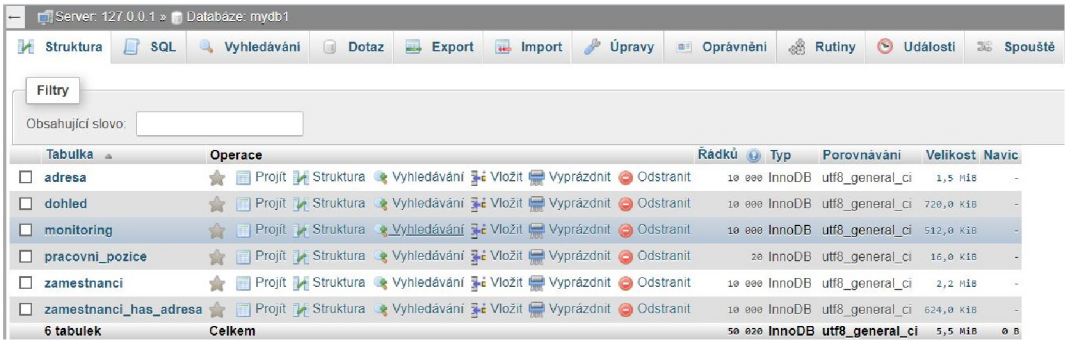
Komunikace s MySQL databází zabrala celkem 46 kB dat. Pro zaslaní příkazu se využilo 1440 bytů. Data, která byla přijata, zabrala celkem 45 kB. Pro přenos bylo využito celkem 75 paketů.

Přenos dat mezí aplikací a Elasticsearch zabral celkem 187 kB. Data, která se na databázi posílala, zabrala celkem 468 bytů, a přijato bylo celkem 186 kB. Pro přenos bylo využito celkem 158 paketů.

Výsledný rozdíl mezi daty, které byly zasílaný z jednotlivých databází je zapříčiněn tím, že MySQL zasílá pouze hodnoty z tabulek, zatímco Elasticsearch posílá celé dokumenty, které obsahují kromě hodnot, také názvy jednotlivých polí.

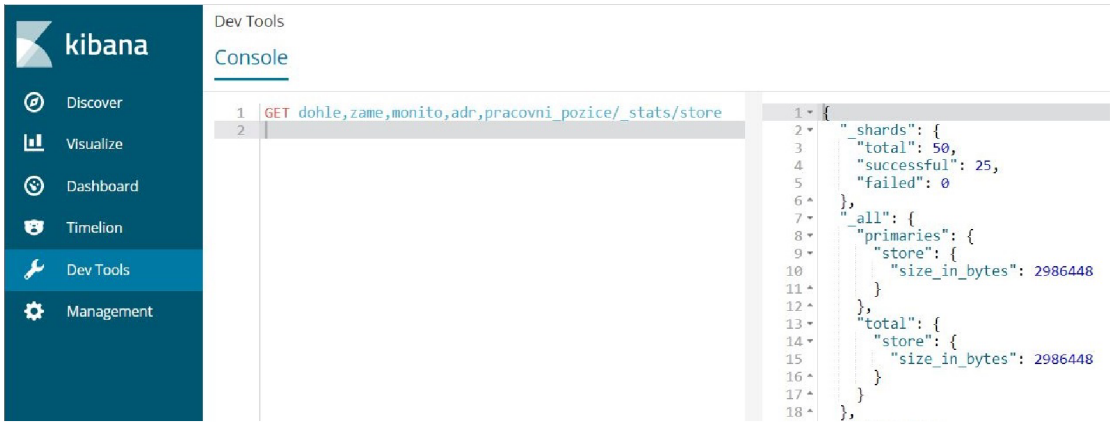
4.4.3 Velikost databází

Tato kapitola se zabývá srovnáním velikostí databází. Obě databáze obsahují stejný počet záznamů. Pro zjištění velikosti databáze MySQL byl použit webový nástroj phpMyAdmin. Kde je možnost zjistit velikost jednotlivých tabulek a celkovou velikost databáze.



Obr. 4.11: Struktura databáze v phpMyAdmin.

Pro zjištění velikosti dat v Elasticsearch byl použit nástroj Kibana a jeho konzole. Příkazem *GET nazev_indexu/_stats/store* se zjistí velikost indexu v bajtech. Pro zjištění velikostí celé databáze, se může za parametr *GET* vypsat všechny indexy, které nás zajímají a Kibana vypíše celkovou velikost dat a velikost jednotlivých indexů.



Obr. 4.12: Konzole v Kibana.

V tabulce 4.1 je vidět, že Elasticsearch zabírá celkem méně datového prostoru. V tabulce *pracovni_pozice*, kde je celkem 20 záznamů, využívá MySQL méně prostoru. Ale v tabulkách, které obsahují více záznamu, tak Elasticsearch využívá datový prostor lépe. Celková velikost MySQL databáze je 5575 kB a velikost Elasticsearch zabírá o celých 1,5 MB méně datového prostoru a to 4030 kB.

	MySQL [kB]	Elasticsearch [kB]
adresa	1500	1214
dohled	720	873
monitoring	515	469
pracovni pozice	16	79
zamestnanci	2200	1393
zamestnanci_has_adresa	624	x
celkem	5575	4030

Tab. 4.1: Srovnání velikosti databází

5 ZÁVĚR

Teoretická část bakalářské práce se zabývá problematikou relačních a nerelačních databází. V první kapitole je popsána relační databáze. Je zde vysvětleno, co je to relační algebra a jaké opera obsahuje. Dále jsou popsány Coddova pravidla, které musí být splněna, aby byla databáze považována za relační. Dále je popsána normalizace a denormalizace databáze. Druhá kapitola se věnovala nerelačním NoSQL databázím. Analyzuje provedení transakcí a CAP teorém, který definuje přístup k databázím pomocí tří ideálních vlastností. Dále popisuje různé typologie nerelačních databází a jejich využití. Třetí kapitola se zaměřuje na dokumentovou databázi Elasticsearch. Představuje základní prvky databáze a vysvětluje provedení operace vyhledávání a mapování. V poslední kapitole je popsán návrh a realizace MySQL databáze, realizace nerelační Elasticsearch databáze a popis aplikace pro měření rychlosti vykonávání dotazů na databázích.

Cílem bakalářské práce bylo vytvoření databáze pro ukládání dat z IoT zařízení monitorující pracovníka v kanceláři. Byl proveden návrh a realizaci relační MySQL databáze, kterou jsem nahrál na server phpmyadmin. Pomocí grafického nástroje Kibana byly dokumenty nahrány do nerelační databáze Elasticsearch. Dále byla vytvořena aplikace v JavaFx, která nabízí možnost spojení s MySQL nebo Elasticsearch a která provede měření. Aplikace měří rychlost provedení příkazů, které jsou v této aplikaci předem definované. Aplikace také nabízí zobrazení výsledků, které vypíše do sloupcového grafu.

Celkem byly provedeny tři typy měření, která porovnávají tyto databáze. První typ měření bylo porovnání rychlosti vykonávání dotazů přes aplikaci. Lepších výsledků zde dosáhla relační databáze MySQL. Pro menší databáze, které obsahují do 1000 záznamů má Elasticsearch zpracování dotazu na samostatné databází pomalejší než MySQL, která zvládne dotaz zpracovat i následně poslat zpátky. U dalšího typu měření bylo zjištěno při stejných dotazech, kolik dat a paketů se využije při posílání. V tomto srovnání dosáhla lepších výsledků databáze MySQL. Tento výsledek je zapříčiněn tím, že MySQL posílá pouze hodnoty, natož Elasticsearch posílá celé dokumenty, které kromě hodnot obsahují také popis jednotlivých polí. U posledního typu měření bylo zjištěno porovnání velikosti samostatných databází. V tomto měření dosáhla lepších výsledků nerelační databáze Elasticsearch. Velikost jednotlivých tabulek a celková velikost je zobrazena v tabulce 4.1. Každá databáze v různých měřeních dosáhla lepších i horších výsledků. Ovšem velkou výhodou databáze Elasticsearch je, že není nutné předem znát strukturu dat, která budou v této databázi uložena.

LITERATURA

[1] OPPEL, Andrew J. *SQL bez předchozích znalostí*. Brno: Computer Press, 2008. ISBN 978-80-251-1707-1.

[2] TYRYCHTR, Jan. *Provozní a analytické databáze: Teoretické základy*. Praha: ČSVIZ, 2015. ISBN 9788087968024.

[3] HARRINGTON, Jan L. *Relational Database Design and Implementation: Clearly Explained*. 3. San Francisco: Elsevier Science, 2009. DOI: 10.1016/B978-0-12-374730-3.X0001-0. ISBN 0123747309.

[4] KENT, William. A simple guide to five normal forms in relational database theory. *Communications of the ACM* [online]. ACM, 1983, 120-125 [cit. 2017-11-29]. DOI: 10.1145/358024.358054. ISSN 0001-0782. Dostupné z: https://dl-acm-org.ezproxy.lib.vutbr.cz/citation.cfm?doid=358024.358054

[5] STEPHENS, Ryan K., Ronald R. PLEW a Arie JONES. *Naučte se SQL za 28 dní: [stačí hodina denně]*. Brno: Computer Press, 2010. ISBN 9788025127001.

[6] Relační a nerelační datový model v kontextu business intelligence. *Webová integrace* [online]. Praha: Ondřej Kopal, 2015 [cit. 2017-10-22]. Dostupné z: http://www.web-integration.info/cs/blog/relacni-a-nerelacni-datovy-model-v-kontextu-business-intelligence/

[7] MARIUS CRISTIAN MAZILU. Database Replication. *Database Systems Journal* [online]. Bucharest Academy of Economic Studies Publishing House, 2010, (2), 33-38 [cit. 2017-12-06]. ISSN 2069-3230. Dostupné z: https://doaj.org/article/7844417b432244a18f931fe71e559b06

[8] HOLUBOVÁ, Irena, Jiří KOSEK, Karel MINAŘÍK a David NOVÁK. *Big Data a NoSQL databáze*. Praha: Grada, 2015. Profesionál. ISBN 978-80-247-5466-6.

[9] POKORNY, Jaroslav. NoSQL databases: A step to database scalability in web environment. *International Journal of Web Information Systems* [online]. 2013, **9**(1), 69-82 [cit. 2017-11-29]. DOI: 10.1108/17440081311316398. ISSN 17440084. Dostupné z: http://www.emeraldinsight.com/doi/full/10.1108/17440081311316398

[10] MONIRUZZAMAN, A B M a Syed Akhter HOSSAIN. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *Journal of Database Theory and Application* [online]. 2013, **2013**(4), 4-7 [cit. 2017-12-06]. Dostupné z: https://arxiv.org/abs/1307.0191

[11] Elasticsearch. *Ludekvesely* [online]. Praha: Luděk Veselý, 2017 [cit. 2017-11-19]. Dostupné z: https://www.ludekvesely.cz

[12] GUPTA, Sheffi a Rinkle RANI. A comparative study of elasticsearch and CouchDB document oriented databases. In: *Inventive Computation Technologies* [online]. Coimbatore, India: IEEE, 2016, s. 1-4 [cit. 2017-11-29]. ISBN 978-1-5090-1285-5. ISSN 1. Dostupné z: http://ieeexplore.ieee.org/document/7823252/#full-text-section

[13] Elastic. *Elastic* [online]. Amsterdam, 2017 [cit. 2017-11-19]. Dostupné z: https://www.elastic.co

[14] KONONENKO, Oleksii, Olga BAYSAL, Reid HOLMES a Michael GODFREY. *Mining Modern Repositories with Elasticsearch*. MSR 2014. New York, NY, USA: ACM, 2014. ISBN 9781450328630.

[15] Database index. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-13]. Dostupné z: https://en.wikipedia.org/wiki/Database_index

[16] WENZEL, Kris. Database Indexes Explained. *Essentialsql* [online]. [cit. 2018-05-13]. Dostupné z: https://www.essentialsql.com/what-is-a-database-index/

[17] BANNON, Ryan. Mysql conceptual architecture. *Technical report, University of Water-100*. 2002, 14.

[18] Logging levels - Logback. *Stackoverflow.com* [online]. stackoverflow, 2011 [cit. 2018-05-13]. Dostupné z: https://stackoverflow.com/questions/7839565/logging-levels-logback-rule-of-thumb-to-assign-log-levels

[19] ANANDHI, R a K CHITRA. A challenge in improving the consistency of transactions in cloud databases-scalability. *International Journal of Computer Applications*. Foundation of Computer Science, 2012, (2), 52.

[20] YEGULALP, Serdar. What is NoSQL? NoSQL databases explained. *Infoworld.com* [online]. 2017, [cit. 2018-05-16]. Dostupné z: https://www.infoworld.com/article/3240644/nosql/what-is-nosql-nosql-databases-explained.html

[21] NoSQL Databases. *Mongodb.com* [online]. MongoDB [cit. 2018-05-16]. Dostupné z: https://www.mongodb.com/nosql-explained

[22] Database Integrity. *Tutorialcup* [online]. Tutorial Cup [cit. 2018-05-16]. Dostupné z: https://www.tutorialcup.com

[23] Why NoSQL Database?. *Couchbase.com* [online]. Mountain View: couchbase [cit. 2018-05-18]. Dostupné z: https://www.couchbase.com/resources/why-nosql

[24] What is Elasticsearch?. *Http://aws.amazon.com* [online]. Seattle: Amazon Web Services, 2017 [cit. 2018-05-19]. Dostupné z: https://aws.amazon.com/elasticsearch-service/what-is-elasticsearch/

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ANSI	Americký národní standardizační institut – American National Standards Institute
BSON	Binárně strukturovaný objektový zápis – Binary Structured Object Notation
CPU	Centrální procesorová jednotka – Central processing unit
CQL	Dotazovací jazyk Cassandra – Cassandra Query Language
DBMS	Systém pro řízení databáze – Ddatabase management system
HTTP	Hypertextový přenosový protokol – Hypertext Transfer Protocol
IoT	Internet věcí – Internet of Things
JSON	JavaScriptový objektový zápis – JavaScript Object Notation
Kb	Jednotka informace – kilobyte
NOSQL	Ne jenom strukturovaný dotazovací jazyk – Not Only Structured Query Language
PHP	Hypertextový preprocesor – Hypertext Preprocessor
RAM	Operační paměť – Random access memory
RDBMS	Systém pro řízení relační databáze – Relational Database Management System
SQL	Strukturovaný dotazovací jazyk – Structured Query Language
URL	Jednotná adresa zdroje – Uniform Resource Locator
UUID	Univerzální unikátní identifikátor – Universally Unique Identifier
XML	Rozšiřitelný značkovací jazyk – Extensible Markup Language

A OBSAH PŘILOŽENÉHO CD

- `/` ..... kořenový adresář přiloženého CD
  - `BP.pdf` ..... bakalářská práce
  - `aplikace_javafx.zip` ..... program javaFx