Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

DIPLOMOVÁ PRÁCE

Využitelnost NoSQL databází v datových skladech

Autor: Bc. Kristýna Sulíková

Studijní obor: im2

Vedoucí práce: Ing. Pavel Čech, Ph.D.

Hradec Králové 2019

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 29. dubna 2019 Podpis:

Poděkování

Ráda bych poděkovala vedoucímu diplomové práce panu Ing. Pavlovi Čechovi, Ph.D. za metodické vedení práce, cenné rady a vstřícné jednání při osobních konzultacích.

Anotace

Diplomová práce se zabývá výzkumem využitelnost NoSQL databází v datových skladech. V teoretické části byly definovány klíčové pojmy jako jsou data, databáze, NoSQL a popis jednotlivých druhů NoSQL databází. Další kapitoly definují pojem datový sklad a Business intelligence. Na základě úvodní rešerše podobných prací byla stanovena metodika praktické části, které spočívá v testování rychlosti odezvy dotazu v SQL databázi Postgre a MongoDB. Na náhodně vygenerovaných datech se provádí testování pro agregační funkce a spojování tabulek. Výsledky slouží k doporučení vhodnosti volby databáze pro určitý typ analytických úloh.

Annotation

This diploma thesis focus on the research of the usability of NoSQL databases in data warehouses. The theoretical part defines key terms such as Data, Databases, NoSQL and the description of individual types of NoSQL databases. Next chapters define the term data warehouse and business intelligence. Initial research of similar works define methodology of the practical part, which consists in testing the response rate of the query in the SQL database Postgre and MongoDB. The randomly generated data is tested for aggregation functions and joining tables. The results will be used for recommend the suitability of selecting a database for a particular type of analysis task.

Obsah

1 Úvod

Problém minulosti byl nedostatek informací, ale dnes je situace opačná. Na trhu dochází k přehlcení informacemi a vlastnit potřebná data v dostupném čase je v zájmu každého ekonomického subjektu. Dle pana Špoka se aktuálně společnost nachází v tzv. postfaktické době, kde se lidé ocitají ve vlastní názorové bublině. Domnívám se, že postfaktická doba se netýká firemního prostředí, které vyžaduje interpretaci reality na základě doložených dat. Firemní prostředí se řadí do kategorie informační společnosti, která zpracovává informace jako významná ekonomická aktiva, díky kterým získá nové příležitosti a činnosti. Dle deníku The Economist není nejcennějším zdrojem na světě ropa, ale data, která jsou označována za ropu digitálního světa. Vznikla nová oblast ekonomiky - datová ekonomika. Data společností se stávají hlavním strategickým zdrojem.

Pro stále rostoucí potřeby a množství zpracovaných dat je technologický pokrok v oblasti informačních technologií nutností. Některé přístupy a architektury, které se využívaly v minulosti nejsou dostačující pro stávající potřeby firemního sektoru. Přicházejí na trh nové technologie a poptávka po kvalifikovaných odbornících vzrůstá. Zvyšují se náklady na informační technologie ve firmách, které mohou mít velmi příznivé ROI.

Trend datového skladu se pohybuje na trhu již několik let. I přes počáteční skepse na časovou a finanční náročnost, jeho zavedení přináší konkurenční výhodu podniků, pro které je důležité business rozhodování založené na reprezentaci dat. Dokazuje to studie společnosti IDC, která analyzovala 62 společností z Evropy a Severní Ameriky. Velikost datových skladů se pohybovala od megabajtů přes 1 terabajt a byly používány déle než 6 měsíců. Statistická analýza prokázala, že průměrná návratnost investic do datových skladů činila 2,3 let [1].

2 Cíl práce a metodika

2.1 Cíl práce

Cílem diplomové práce je nalezení využitelnosti NoSQL databází v datových skladech. Datový sklad není chápán pouze jako úložiště dat, ale praktická použitelnost spočívá v získávání informací z dat, které lze dosáhnout pomocí Business Intelligence nástrojů.

V datových skladech se nejčastěji provádějí operace nad daty analytického rázu. Využitelnost NoSql by tedy měla být ve schopnosti zpracovávat a tvořit tyto dotazy. Pro zkoumání byly zvoleny agregační dotazy a spojování tabulek pro řešení vazby 1:N. Hodnotícím kritériem je rychlost, a proto byla testována hypotéza: NoSQL DB vrací data v rychlejším čase než SQL databáze. Diplomová práce si klade za cíl zkoumat vliv růstu počtu dat z hlediska času pro dotazy z předchozího měření.

V diplomové práci bude řešena situace, jakým způsobem převést NoSQL databázi na SQL. I v tomto případě bude provedeno zkoumání stejných dotazů a hypotéz, jako v předchozím testování.

Bez ohledu na výsledek hypotézy praktické použitelnosti NoSQL databází jako zdroj pro datové sklady, si práce klade za cíl zkoumání aktuální situace na trhu s NoSQL řešeními a jejich reálné využití.

2.2 Metodika

V teoretické části práce byla využita analýza základních pojmů, které souvisí se zkoumanou problematikou. Byl proveden průzkum trhu aktuálních NoSQL databázových řešení a zástupci z každého typu databáze byly krátce popsány.

Po prozkoumání teoretického základu a úvodní rešerše se v praktické části zkoumají rozdíly mezi relační a nerelační databází na základním vzorku dat. Data byla vygenerována automatickým náhodným generátorem, tak, aby byly vhodně použity pro SQL i NoSQL. Nad daty byly provedeny dotazy obsahující agregační funkce a dotazy se spojením více tabulek.

Vhodným statistickým zpracováním by byl víceúrovňový test, ale vzhledem ke komplexnosti tohoto testování byl v diplomové práci aplikován jednostranný parametrický Studentův T-test. Byla zvolena nulová hypotéza. Na hladině významnosti = 0,05 lze testovanou hypotézu vyvrátit či potvrdit.

Pro zkoumání časové odezvy dotazů při vlivu růstu počtu dat byl vzhledem k rozsáhlosti měření používán aritmetický průměr z naměřených běhů. Celkové výsledky byly znázorněny graficky pro snadnou reprezentaci časových rozdílů.

Měření rychlosti odezvy bylo provedeno i pro konektor, který napojil NoSQL databázi na SQL. Výsledky byly zkoumány pro základní množinu dat, tak aby ukázaly rozdíly v odezvě pro dotazy přímo z NoSQL versus konektor.

3 Rešerše podobných prací

Pro zahájení výzkumu diplomové práce byla provedena rešerše prací, které se zabývají podobnou problematikou jako tato diplomová práce.

První diplomová práce [2] se zabývala analýzou NoSQL databází a jejich srovnání. V práci byly zkoumány základní kategorie NoSQL databázových systémů, tedy databáze typu klíč - hodnota, sloupcová databáze, dokumentová databáze a grafová databáze. Metodika zkoumání probíhala ve dvou rovinách. Prvním typem zkoumání bylo porovnání databází dle funkcionality, partitioning a podpora programovacích jazyků. V druhé části došlo k porovnání databází za pomocí technologie YCSB, která vyhodnocuje výkon počítačových programů. Technologie obsahovala 6 základních testů, které byly provedeny nad databázemi Redis, Apache Cassandra a MongoDB. Nejlepších výsledků dosáhla databáze MongoDB. Mimo naměřených výsledků jsou popsány základní principy fungování, postup instalace a zhodnocení práce s databázemi.

Druhá diplomová práce [3] se zabývala srovnáním relačních, NoSQL databází a OLAP. Cílem byla analýza přínosu čtyř základních NoSQL databází a jejich využití pro analytické účely. Výsledek práce poukazuje na fakt, že nejvyšší užitek pro zpracování dat a analýzu je kombinace obou přístupů řešení. Jelikož NoSQL databáze nejsou vhodné pro pokročilé analytické funkce. Dosáhnout komplexních analytických výsledků lze za pomocí specializovaných analytických nástrojů. Při srovnání s architekturou BI řešení jsou NoSQL databáze doplňkem datových skladů, datových tržišť i OLAP databází.

Třetí výzkumná práce [4] si klade velice podobné cíle práce, jako tato diplomová práce. Zkoumá výkonnost databáze MongoDB a Microsoft SQL Server z hlediska časového běhu. Celkem bylo testováno 7 dotazů. V této práci byly testovány agregační funkce v

MongoDB s využitím principu MapReduce. Výsledky ukázaly na lepší výkon MongoDB při ukládání, aktualizaci a jednoduchých dotazů. Ale v případě agregačních dotazů a dotazování pro ne-klíčové hodnoty bylo SQL řešení z hlediska výkonu lepší.

Čtvrtá výzkumná práce [5] se zabývá zkoumáním adaptérů, které umožňují implementaci transformaci dat z MySQl do HBase. Podobně, jak je tomu v této diplomové práci, je zde použit JDBC ovladač pro MySQL jako RDB konektor. Empirické zkoumání probíhá ve třech režimech přístupu k dotazování, kde je vyhodnocena výkonnost a analýza výsledku. Mezi dotazovací přístupy patří blokování transformačního režimu, režimu výpisu a režim přímého přístupu k datům. Výsledky ukázaly, že nejdelší doba zpracování je při transformačním režimu a naopak v přímém přístupu k datům je doba nejkratší. V experimentech výsledek přímo ovlivňuje i nárůst objemu dat.

Pátá výzkumná práce [6] se zabývá křížovou databázovou analýzou MySQL a NoSQL databází MongoDB a Cassandra z hlediska energetické účinnosti a výkonnosti. V práci se aplikují optimalizační techniky i včetně indexování. V metodice pro měření se používají Cloud Server Benchmark YCSB. Výsledky ukázaly, že za použití indexů se MongoDB zrychlí a sníží se spotřeba energie. V případě Cassandry díky ukládání dat do mezipaměti řádků napomáhá výkonu i energetické účinnosti. Avšak i přesto jsou v těchto databázích značné rozdíly.

V této diplomové práci je užší výběr použitých databází a cíl je zaměřen na rychlost odezvy a praktickou využitelnost MongoDB. Nejsou zde řešeny ani zkušenosti s prací v NoSQL databázích, jelikož to považujeme za osobní názor. Na rozdíl od páté výzkumné práce nepoužíváme indexovaná data.

4 Data, informace a znalosti

4.1 Data

Základní úlohou dat je reprezentovat reálné objekty a popisovat realitu. Data představují formalizovaný záznam lidského poznání pomocí symbolů (znaků), které umožňují přenos, uchování, interpretaci a zpracování. Smysluplné informace pak vznikají v procesu interpretace dat člověkem [7].

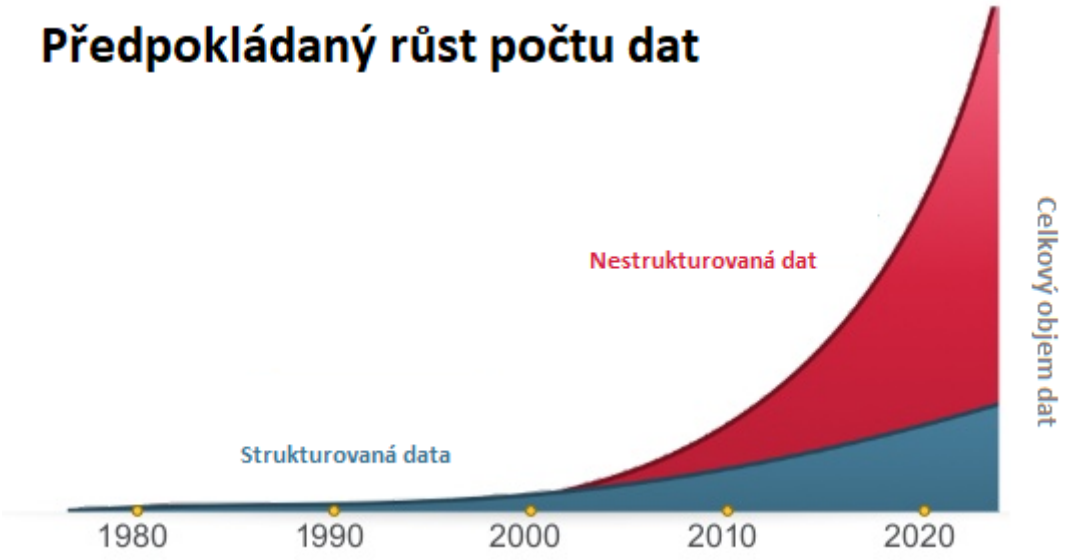
Diplomová práce se bude zabývat daty v kontextu počítačové vědy, kde mají data následující členění:

- STRUTUROVANÁ DATA - popisují strukturu dat ve formě relační databáze. Mají předem definovanou strukturu a explicitně vyjadřují kontext. Lze je dobře ukládat, zpracovávat a analyzovat. Stávají se jedním z doporučení pro internetové vyhledávače, kde se pojem strukturovaná data mnohdy zaměňují s pojmem mikro data.

- NESTRUKTUROVANÁ DATA - na rozdíl od strukturovaných dat nemají přesně definovanou strukturu. Do této kategorie spadají obrázky, videa, internetové komunikace, údaje ze senzorů či informace o nákupním chování zákazníků. Vytěžení informací z nestrukturovaných dat přináší pro firmy velmi ceněné informace a velkou konkurenční výhodu. Jejich hlavním přínosem je učení se nad nestrukturovanými daty a hledání různých vzorců chování pomocí umělé inteligence.

- SEMISTRUKTUROVANÁ DATA - jsou považovány za průnik předchozích kategoriích. Jejich struktura se mění nepredikovatelným způsobem.

S růstem technologického pokroku rostou i data. Jejich předpokládaný nárůst je vidět na obrázku 4.1. Objem nestrukturovaných dat roste oproti strukturovaným několikanásobně a v budoucím vývoji budou tvořit až 80 % z celkového počtu dat [8]. Příčinou velkého růstu je popularita sociálních sítí, které se staly součástí lidského života.



Obrázek 4.1: Trendy ve vývoji dat [8]

Data v organizaci představují podniková aktiva, která se spravují zodpovědně a průběžně způsobem, který lze auditovat. Vedení podniku data využívá jako základní zdroj pro rozhodovací proces. Zda management informace použije vhodně či nevhodně, je zcela jiná otázka.

4.1.1 Big data

Zpracování příliš velkých objemů dat vede k zavedení nových přístupů systémů zpracování dat. S příchodem nové éry webu 2.0 vzrostl objem dat, tzv. Big data. Do této kategorie spadají strukturovaná, nestrukturovaná a semi-strukturovaná data. Big data mají složitou povahu, které vyžadují silné technologie a pokročilé algoritmy pro jejich zpracování.

Pojem Big, neznamená výhradně velký objem dat, ale převážně zpracování a vizualizaci dat v reálném čase z různých zdrojů. Ani sečtení bilion dat v Excelu nevystihuje

pojem Big, jelikož se jedná pouze o jednu operaci. Big data je sada technologií, která umožňují masivní paralelní zpracování dat. Big data se vyznačují charakteristikami, zvanými 3V:

- Objem (volume) - velké objemy dat jsou průběžně generovány z různých typů zařízeních. Podle zprávy IDC objem dat do roku 2020 dosáhne 40 bajtů Zeta a zvýší se až 400 krát
- Rychlost (velocity) - data jsou generována vysokou rychlostí, proto se klade velký důraz na rychlé zpracování. Dobrým příkladem zpracování Big dat je YouTube.
- Rozmanitost (variety) - mnoho formátů a distribuce z různých zdrojů úplných či neúplných dat

Charakteristika 3V se doplňuje o další důležité body: účel (vision), validace (validation), vizualizace (vizualization) a rozmanitost (varibaility) [9]. Společnost Gartner tyto V časem rozšiřuje, i přesto pro některé firmy není pojem Big data jednoznačně definován.

V praxi se s pojmem Big data setkáváme například při implementaci inteligentní sítě, která řídí spotřebu elektřiny v reálném čase a sleduje provoz inteligentních sítí. Generování zdravotních dat z heterogenních zdrojů a následná personalizace zdravotnických služeb.

S pojmem Big data setkáme s aktuálním fenoménem zvaným IoT [10]. Největší využití Big dat je v oblasti zdravotnictví, finanční sféře a v e-commerce tvrdí Big Data Science Leader Dagmar Bínová z firmy Adastra [11].

4.1.2 Metadata

Kolektiv pod vedením Marka Humphriese v knize Data warehousing uvádí: Metada jsou již tradičně definována jako data o datech [12]. Autoři zde popisují netadata, jako určitou formu abstrakce, která popisuje strukturu a obsah datového skladu. Na rozdíl od běžných dat jsou metadata strukturovaná do určité míry a jsou shromážděna, aby splnila daný účel.

Metadata jsou uložena v relačních tabulkách jako pole, respektive záznam. Jejich tvorba spočívá v kódování, nejčastěji formou XML struktury, která se načítá přímo z externích zdrojů nebo v jiných případech se generují prostřednictvím uživatelského rozhraní softwaru.

Dalším způsobem kódování metadat je pomocí RDF, který na rozdíl od XML stromové struktury umožňuje modelování za pomocí grafu pro sémantický web. Metadata se stávají klíčovým problémem sémantického webu, proto nelze považovat definici, že metadata jsou data o datech, jako konečnou. [13].

V diplomové práci se budeme setkávat s pojmem metadata v oblasti datových skladů, kde hrají hlavní roli pro jejich užitečnost pochopení datových položek. Metadata zajišťují snadnější tvorby podnikových analýz a následnou správnou interpretaci dat na základě definování datových formátů. Jejich hlavní výhody jsou shrnuty do následujících bodů:

- Zvyšují a udržují kvalitu dat
- Obsahují popis pravidel pro opravu chyb
- Popisují transformaci zdrojových dat
- Obsahují informace o původu dat

Obrázek 4.2: Typy metadat[14]

5 Databáze

Nevýhodou hromadného zpracování dat a souborů v minulosti vedly k novému přístupu - tzv. databázovému, jehož definicí je odtržení údržby dat od uživatelských programů. Data již nejsou organizovaná v izolovaných souborech, ale v komplikované centrálně zpracované struktuře zvané databáze. Databáze je cokoliv - jednoduchá množina dat, jako je například uživatelský seznam nebo složitá množina různých nástrojů serverové architektury.

Systém řízení bází dat SŘBD je speciální programové vybavení, které organizuje centrální správu databáze. Ve vzájemné vazbě s určitým databázovým systémem (DBS) tvoří komplexní databázové systémy, které umožňují organizovat operativní data v podnicích. Každý informační systém se skládá z databáze a následujících komponent:

- Datové prvky
- Vztahy mezi prvky dat
- Integritní omezení
- Schéma

Části databáze se výhradně neskládají z dat zpracovaných vhodným softwarem, ale i systematické metody návrhu databáze, které splňují předem daná kritéria. Cílem tedy není pouze optimalizace vlastní technologie databáze, ale i architektura návrhu rozsáhlé databázové struktury[15].

5.1 Databázový model

Schéma pro strukturovaná data v databázi se nazývá databázový model. Pojem databázový model zavedl C. J. Date, který ho popsal jako definici entit, atributů a integritních omezení entit [16]. Model by měl poskytovat soubor abstrakcí pro modelování informací, které slouží ke konkrétnímu účelu a dále poskytovat soubor pravidel pro uchování, indexování a načítání dat. Výsledkem databázového modelování je popis struktury dat. Existuje několik druhů databázových modelů:

5.1.1 Síťový model

Základním principem síťového modelu je ukládání dat do záznamů, které se skládají ze skupin datových položek spojujících konkrétní případy jednotlivými vazbami. Na rozdíl od hierarchického modelu umožňuje vytváření většího počtu vazeb k více entitám. Definuje pouze binární vztahy typů 1:1 či 1:N mezi dvěma typy záznamů.

V současné době se síťový model nepoužívá, jelikož princip síťových modelů je již zastaralý a nahradil ho právě princip relační.

5.1.2 Hierarchický model

Hierarchický model se zaměřuje na hierarchii vztahů mezi objekty a je charakteristický stromovou strukturou. Hlavním principem je volba jednoho hlavního prvku (rodiče), kterému se přisuzují další pod prvky (děti). Tato struktura umožňuje duplicitní informace pro nižší segment následníka.

Z formálního hlediska je hierarchický model považován za speciální typ síťového modelu. Dnes bychom mohli spatřit tento model v HFS od IBM.

5.1.3 Relační model

Tvoření databáze dle relačního modelu patří mezi nejoblíbenější techniky. Hlavní charakteristikou relačního modelu je reprezentace pravidelně uspořádaných struktur s řádky a

sloupci, které se nazývají relace. Jeden význam v databázi připadá na jednu hodnotu řádku a sloupce. Oproti předchozím modelům se zde vyskytují vazby 1:1 a 1:M.

Relační model klade velký důraz na zachování integrity prostřednictvím primárního klíče, cizího klíče, referenční integrity a normálního tvaru. Výhodou relačního modelu je realizace nepředpokládaných vazeb při první fázi návrhu RDBMS. S příchodem relačních databází vznikl dotazovací jazyk SQL. Dnes jsou relační modely považovány za standard v oblasti tvorby datových skladů a dolování v datech.

5.1.4 Objektový model

Při prvním návrhu objektového modelu měl být hlavní účel poskytnutí perzistence pro objektově orientované programovací jazyky, které byly na vzestupu a měly nahradit strukturovaný přístup. Programový kód byl lépe pochopitelný a snížilo se jeho množství. Dotazovacím jazykem jsou zde objektově orientované programovací jazyky. Charakteristikami objektového modelu jsou třídy s atributy a metodami, integritní omezení, podpora zapouzdření, násobná dědičnost a abstraktní datové typy.

5.1.5 Objektově relační model

Objektově relační model je synonymem pro modely sjednocující relační a objektové principy databází (ORDBMS). Pro zjednodušení přidávají vlastnosti objektového přístupu do tabulek a umožňují tak tvorbu ad hoc dotazů, které v případě objektového modelu nebyly příliš vhodné.

Hlavní výhodou je použití při větším objemu dat, právě kvůli vyjádřením složitých vztahů mezi objekty. Dotazování se provádí prostřednictvím SQL3.

NoSQL modely jsou vysvětleny v kapitole 5.3.4 , jelikož každá NoSQL databáze má rozdílná specifika modelu dle typu použití.

5.2 SQL

SQL je nejrozšířenější strukturovaný dotazovací jazyk, který se používá pro manipulaci a získávání dat z relačních databází. Jak z definice vyplývá, je příkazově orientovaný a řadí se mezi neprocedurální neboli deklarativní jazyky. Syntaxe se nepatrně odlišuje vzhledem k volbě relační databáze. Procedurální rozšíření jazyka nabízí Oracle v podobě PL/SQL (Procedural language/SQL) nebo Microsoft v podobě Transact-SQL. Příkazy jazyka SQL rozdělujeme do následujících kategorií a stručného přehledu konkrétních příkazů [17]:

- DDL (Data Definition Language) - CREATE, ALTER, DROP
- DQL (Data Query Language) - SELECT
- DML (Data Manipulation Language) - INSERT, UPDATE, DELETE
- DCL (Data Control Language) - GRANT, ALTER
- Příkazy řízení transakcí - COMMIT, ROLLBACK

  Zpracování příkazů není ve stejném pořadí zápisu příkazu.

5.2.1 ACID

Relační model je vhodný pro zpracování transakcí. Vyznačuje následujícími typickými vlastnostmi transakcí, známé pod zkratkou ACID [17] :

- Atomicita - transakce proběhne správně pouze za předpokladu, že každá její část obsahuje správné prvky. V případě chyby se neprovede žádná transakce.
- Konzistence - transakce proběhne za předpokladu neporušení integritního omezení.
- Izolace - změny uvnitř transakce se projeví až po jejím konečném běhu.
- Trvalost - uložená transakce v databázi se neztratí.

5.2.2 Agregační funkce v SQL

Agregační funkce jsou označovány za seskupovací funkce, proto jsou vždy definovány s konstrukcí GROUP BY nebo HAVING. Na základě hodnoty v daném sloupci vracejí konkrétní požadovanou hodnotu.Agregují všechny řádky do jedné hodnoty. Standard ANSI definuje prvních pět agregačních funkcí:

- COUNT - součet počtu hodnot uvedených v podmínce WHERE
- SUM - součet hodnot uvedených v podmínce WHERE
- AVG - průměr hodnot z daného sloupce
- MAX - nejvyšší hodnota v daném sloupci
- MIN - nejmenší hodnota v daném sloupci

Agregační funkce pracují s číselnými hodnotami a výsledkem je také číselná hodnota.

5.2.3 Trendy a budoucnost SQL

Z historického hlediska je SQL starším jazykem, nástroje podporující SQL se nevyhnou trendům a růstu nových požadavků. Například v oblasti bezpečnosti na úrovni řádků a maskování dynamických dat navrhnul SQL Server 2017 rozšíření. Vzhledem k stále rostoucímu trendu cloudových aplikací se stává platforma Azure budoucností Microsoftu. Ani oblast umělé inteligence není pozadu a Microsoft s platformou Cloud AI označuje SQL server za "první RDBMS se zabudovaným AI". Jedná se o nástroje podporující začlenění strojové učení a knihoven napsaných v R nebo v Pythonu [18].

5.3 NoSQL

Vznik NoSQL se traduje roku 1998. Zkratka představuje pojem, který vylučuje SQL, ale správná definice je "nejen SQL". Hlavní myšlenkou je spojení těchto dvou technologií dohromady [19]. NoSQL řešení nevyužívá přímo SQL pro práci s daty a jsou vhodné pro zpracování velkých objemů dat, u kterých je důležitější výkon než konzistence.

Prozatím jsou NoSQL databáze vnímány jako vysoce výkonné datové struktury vhodné pro filtrování a vyhledávání.

5.3.1 BASE

NoSQL řešení považuje ACID přístup za příliš přísný a preferuje pojem BASE, který se skládá z následujících bodů [20]:

- Dostupnost - data v NoSQL databázi jsou uložené s vysokým stupněm replikace, který umožňuje v případě poruše spojení mezi uzly zachovat funkčnost po celou dobu.
- Degradace - stav systému se mění v čase. Vzniká nekonzistence tedy není problémem databázového modelu, ale funkci backend.
- Nahodilá konzistence - konzistentní data jsou dosahována postupně v budoucnosti. Není zaručeno, kdy ke konzistenci dojde. Pravý opak principu ACID.

5.3.2 CAP teorém

Zajištění ACID vlastností je nákladné či zbytečně obtížné v případě distribuce, replikace a výpadcích spojení. Proto jsou mimo principu BASE kladeny na NoSQL databáze požadavky splňující vlastnosti CAP teorému. V ideálním případě by došlo ke splnění těchto tří vlastností:

- Konzistence (Consistency) - existuje jediná aktuální verze dat.
- Dostupnost (Availability) - všechny požadavky jsou systémem zajištěny.
- Odolnost vůči rozpadu sítě (Partition tolerance) - zajištění funkčnosti při rozpadu několika izolovaných částí.

Bylo dokázáno, že lze dosáhnout pouze dvou vlastností. Tvrzení a samotný CAP teorém publikoval Erik Brewer v roce 2000. Cílem dnešního CAP teorému je maximalizovat konzistenci a dostupnost [21].

5.3.3 Vztah 1:N

Neexistuje pouze jedna cesta pro řešení vztahu 1:N mezi objekty, jak je tomu v SQL. Existují 3 možnosti, které jsou použitelné při návrhu NoSQL designu [22].

1. One-to-Few

   Pro získání vnořených informací nemusí být proveden samotný dotaz, jelikož data jsou již uložena v rámci dokumentu. Tato možnost je vhodná pouze pro malý počet záznamů.

2. One-to-Many

   V tomto případě je nutné pro získání dat spustit dva separátní dotazy, jelikož daný dokument obsahuje pouze ID vnořeného dokumentu. Tato možnost je vhodná pokud je k dispozici tisíce záznamů.

3. One-to-Squillions

   Milion záznamů, ani by nebylo technicky realizovatelné zvolit jinou vazbu.vPro získání dat z tohoto typu spojení jsou zapotřebí také dva dotazy, ale je odlišné referencování mezi dvěma kolekcemi. Ve vztahu rodič - potomek si drží informaci pouze potomek, ke kterému náleží rodič. Pokud databáze obsahuje milion záznamů, není jiná možnost, než použít tento způsob vazby 1:N.

5.3.4 Kategorie NoSQL

Kategorie NoSQL databází se rozdělují podle datových modelů s odlišným přístupem ke zpracování dat. Každá kategorie má několik představitelů, které jsou většinou opensource aplikace. Základními kategoriemi jsou [23]:

- Key-value databáze

  Key-value databáze ukládají a načítají data jako klíč-hodnota, kde klíč je jedinečný identifikátor. Každá uložená hodnota představuje kompletní souhrn dat. V tomto modelu nejsou uloženy žádné vztahy a proto jsou hodnoty uloženy jako řetězce, celá čísla či seznamy. V databázi Key-value se nepracuje s komplexními dotazy. Ty jsou zpracovány na úrovni aplikační logiky. Příkladem databáze tohoto typu je Redis, Cassandra nebo Voldemort, které se používají v oblasti sociálních sítí.

- Dokumentová databáze

  Databáze dokumentů má podobné charakteristiky jako Key-value databáze, tedy umožňuje ukládání volné strukturované sady klíčů-hodnot v dokumentech. Dynamické schéma je definováno kolekcemi. Mezi představitelé se řadí MongoDB nebo CouchDB.

- Databáze grafů

  Grafové databáze nevyžadují formální schéma, ale vyžadují formální strukturu, která se skládá z vrcholů a hran. Data se tak získávají pomocí modelování objektů uzlů a hran. Uzel představuje entitu ukládající a získávající atributy a hrana je vlastností mezi jedním nebo dvěma uzly. Oproti ostatním kategoriím podporují primitivní datové typy. Nejznámnější představitel je Neo4j.

- Sloupcová databáze

  Sloupcově orientovaná databáze ukládá data ve sloupcové formě a tím se tak výrazně liší od relačních databází, jelikož data jsou organizována do vícerozměrné mapy. Sloupec je základní jednotka, která se skládá z názvu a hodnoty. Řádky mají jedinečný identifikátor a jeden či více sloupců, a tím tak tvoří dvouúrovňovou mapu.Mezi zástupce se řadí Hbase.

5.3.5 NoSQL nástroje

MongoDB

MongoDB se řadí mezi dokumentové databáze, které ukládají data do flexibilních dokumentů typu JSON a je v jádru distribuovaná. Struktura dat se tak v čase mění a dochází k mapování objektů v kódu aplikace. Analytické dotazování se provádí pomocí ad hoc dotazů a agregace dat v reálném čase poskytuje výkonný způsob přístupu k datům.

MongoDB je open-source, který je dostupný ve dvou verzích Community edition a Enterprise. Pro programátory existuje nástavba zvaná mongo Shell, která umožňuje používání databázové instance MongoDB psanými příkazy v JavaScriptu [24].

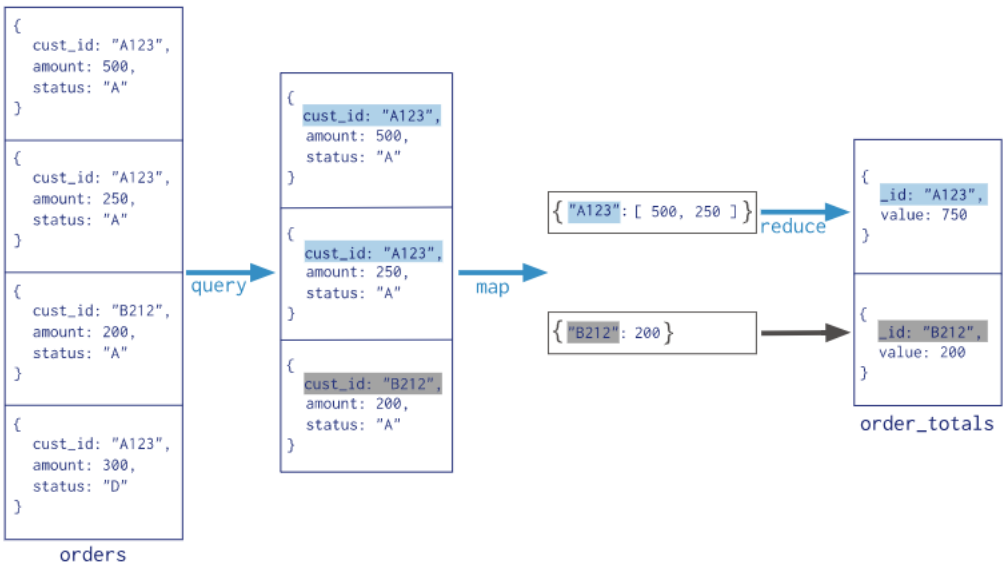
Map reduce v MongoDB

Map reduce je paradigma pro zpracování velkých objemů dat do smysluplných agregovaných výsledků. Je to základní agregační operace, která používá dvě funkce Map a Reduce. Pomocí funkce Map můžeme seskupit určité prvky ze vstupu, případně vstup upravit, přičemž výsledkem funkce Map je struktura key-value. Pokud již klíč existuje, tak MongoDB automaticky tyto dvě data spojí. Reduce nám vezme vstupy z předchozí fáze Map a může je agregovat dle dané poskytnuté funkce. Obě dvě funkce jsou JavaScriptové funkce [25]. Názorný přehled lze spatřit na obrázku 5.1

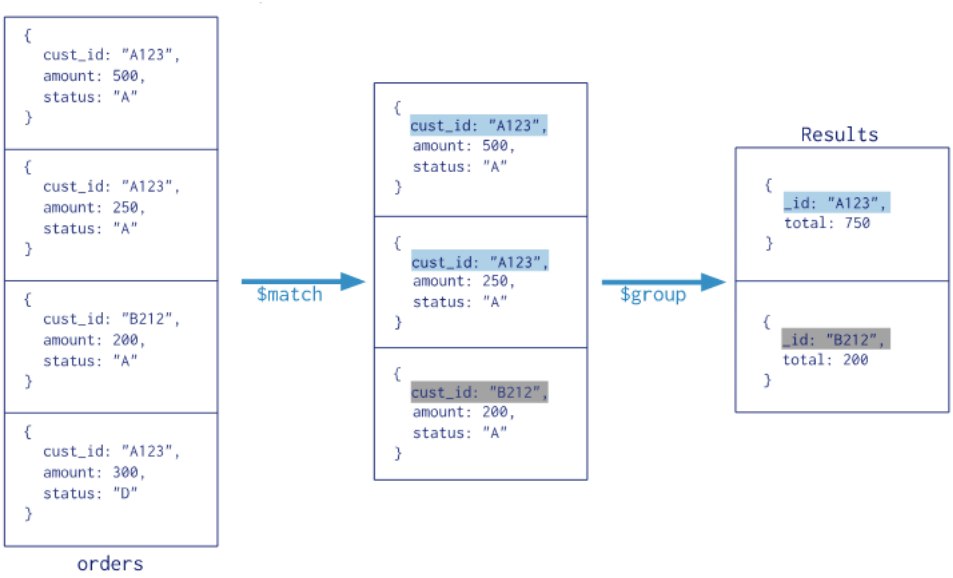
Tímto způsobem jsme schopni dosáhnout jakékoliv agregace, ale v případě komplexního spojení dat může působit nepřehledně a formulování bude velice komplikované. Dle oficiální dokumentace MongoDB je vhodnější použití pro dotazování princip Agregačních pipelines, který vychází z principů Map reduce.

Agregační pipelines v MongoDB

Agregační pipelines utvářejí rámec pro agregaci dat a pro zpracování dat transformuje dokumenty na agregované výsledky. Data se zpracovávají v kolekcích a při generování dochází k tvorbě nových dokumentů nebo filtrování stávajících dokumentů. Porovnáním obrázků 5.1 a 5.2 se jeví agregační pipelines jako zjednodušené řešení pro psaní kódu dotazu.



Obrázek 5.1: Map-Reduce v MongoDB [25]



Obrázek 5.2: Agregační pipelines v MongoDB [26]

Při používání tohoto principu můžeme narazit na limity agregačních pipelines, které spočívají v limitech paměti RAM.

Redis

Redis je představitelem key-value databáze a stejně jako MongoDB je open-source. Podporuje datové struktury v podobě řetězců, seznamů sad, bitmapy nebo geoprostorové indexy. Umožňuje seskupení transakcí tak, aby byly provedeny jako jeden dotaz. Pro rekonstrukci dat po restartu paměti v systému se data ukládají v nečitelném formátu na disk [27]. Dle obrázku 8.1 je druhou nejoblíbenější NoSQL databází.

Cassandra

Cassandra je stejně jako Redis key-value databáze, která je open-source. Jedná se o distribuovaný systém navržený za účelem nasazení velkého počtu uzlů napříč několika datovými centry. Dostupnosti a odolnosti vůči přerušení dává přednost před konzistencí. Dotazování je za pomocí CQL (Cassandra Query Language) [28].

Neo4j

Neo4j je zástupcem grafové databáze, která nativně ukládá a zpracovává data v podobě grafů. Skládá se z uzlů a vztahů, které jsou komponentami spojenými vlastnostmi. Uzly a vztahy dohromady tvoří prvky s jednoduchými nebo složitými cestami. Je implementována v jazyce Java a dotazování probíhá pomocí jazyka Cypher, který je považován za jednoduchý a logický jazyk pro uživatele [29].

5.4 SQL versus NoSQL

Na základě předchozího textu byly zaznamenány rozdíly mezi SQL a NoSQL řešením. Databáze SQL poskytují úložiště souvisejících datových tabulek, která vytváří přísnou datovou šablonu. Oproti tomu NoSQL ukládají hodnoty ve formátu XML či JSON, jsou tak flexibilnější a umožňují přidání dat bez předem specifikované kolekce, to ale vede k problémům s konzistencí.

V případě SQL data normalizujeme, ale při tvorbě DWH dochází k denormalizaci, podobně jak je tomu v případě NoSQL. Denormalizací se zabraňuje používání příkazu JOIN, který zpomaluje dotaz. NoSQL se tak stávají rychlejším řešením. Avšak nevhodně navržená a nastavená NoSQL databáze přináší větší problémy, než správně navržená SQL databáze. Problém tak nastává ve znalostech řešitelského týmu, kterému chybí potřebné znalosti k zavedení NoSQL řešení.

SQL řešení je vhodné pro projekty, kde je zásadní integrita dat, standard a mnoholeté zkušenosti vývojářů pro logicky související požadavky na diskrétní údaje. Například elektronický obchod vyžadující robustní podporu transakcí. NoSQL řešení je vhodné pro projekty, kde se požadavky vyvíjí v čase a škálovatelnost spolu s rychlostí jsou nezbytnými atributy.

6 Datové sklady

6.1 Historie a vývoj datového skladu

V roce 1990 Bill Inmon získal titul zakladatele datového skladu, tak se stal nejvýznamnějším představitelem této problematiky. Dalším významným představitelem a průkopníkem datových skladů byl Ralph Kimball. Oba dva se na problematiku datových skladů soustředili z jiného úhlu pohledu a vznikly tak dva základní přístupy, které se používají dodnes.

1. Model dle Inmona

   Model dodržuje vývojový přístup shora dolů, který přízpůsobuje tradiční relační databázové nástroje vývojovým potřebám celopodnikového datového skladu. Z tohoto celopodnikového úložiště jsou tvořeny jednotlivé oddělené databáze, které slouží pro potřebu rozhodování.

2. Model dle Kimballa

   Model představuje modelování dat a jejich rozměrů. Upřednostňuje přístup zdola nahoru. Nebuduje jednotnou celopodnikovou databázi, ale navrhuje založení jedné databáze pro každý důležitý obchodní proces.

Tyto dva modely patří k základním principům při tvorbě datového skladu a je důležité zvolit si jeden z těchto přístupů [30]. Jejich vzájemné porovnání se nachází na obrázku 6.1.

Charakteristiky	Kimball	Inmon
Podporu rozhodování	taktické	strategické
Integrace dat	Individuální obchodní jednotky	Celý podnik
Struktura dat	KPI a obchodní měřítka	různorodá nespojitá data
Perzistence	Stabilní	Výkyvy
Velikost týmu	Malý tým odborníků	Velký tým specialistů
Časové omezení	Naléhavé odpovědi nad datovým skladem	Delší časová odezva je přijatelná
Pořizovací náklady	Nízké	Vysoké

Tabulka 6.1: Modely datových skladů [31]

6.2 Definice datového skladu

Datový sklad je kolekce sjednocených, předmětově orientovaných databází navržených za účelem poskytovat požadované informace pro podporu rozhodování [32]. Metodika dimenzionálního modelování se stala standardem v systémech na podporu rozhodování. Koncept datového skladu je zpravidla nezávislý na platformě, nicméně převládající technologií jsou řešení založené na relačních databázích.

Struktura pro analýzu dat je odlišná od struktury transakčních systémů. V transakčních databázích jsou data neustále měněna, oproti datovému skladu, kde jsou data statická. Na základě tohoto faktu se vyvíjel princip datových skladů, který byl původně navržen jako oddělené komponenty architektury.

Datový sklad integruje masu surových dat z primárních systémů, provozních systémů a externích zdrojů. Díky této integraci se stává základním pilířem pro získávání informací pro dlouhodobé rozhodování, plánování a tvorbu podnikových analýz [12].

Hlavní výhody datového skladu spočívají v [33]

- umožňuje přístup k integrovaným a konzistentním datům
- ukládá data ve vhodné struktuře pro podporu rozhodování
- izoluje OLTP databáze od operací pro podporu rozhodování
- příležitost pro poznání dat
- konkurenční výhoda pro společnost

Tvorba datových skladů přináší i několik negativních stránek. Vývoj datových skladů není jednoduchá a levná záležitost. Nedostatek znalosti a pochopení dat či zajištění její kvality vede k nevhodnému návrhu a používání datových skladů. S tím souvisí riziko velikosti a rostoucího objemu dat. Nevhodné zvolení architektury, která není dostačující pro objem dat, znamená růst nákladů na rozšíření a provoz datového skladu.

Proces sjednocení dat do datového skladu přináší problémy, které by se měly v rámci implementace ošetřit. Jedná se například o redundantní data z různých databází. Častým problémem je formát dat a sjednocení odlišných měn. Tyto problémy se řeší na úrovni návrhu architektury datových skladů, která je popsána v následujících kapitolách.

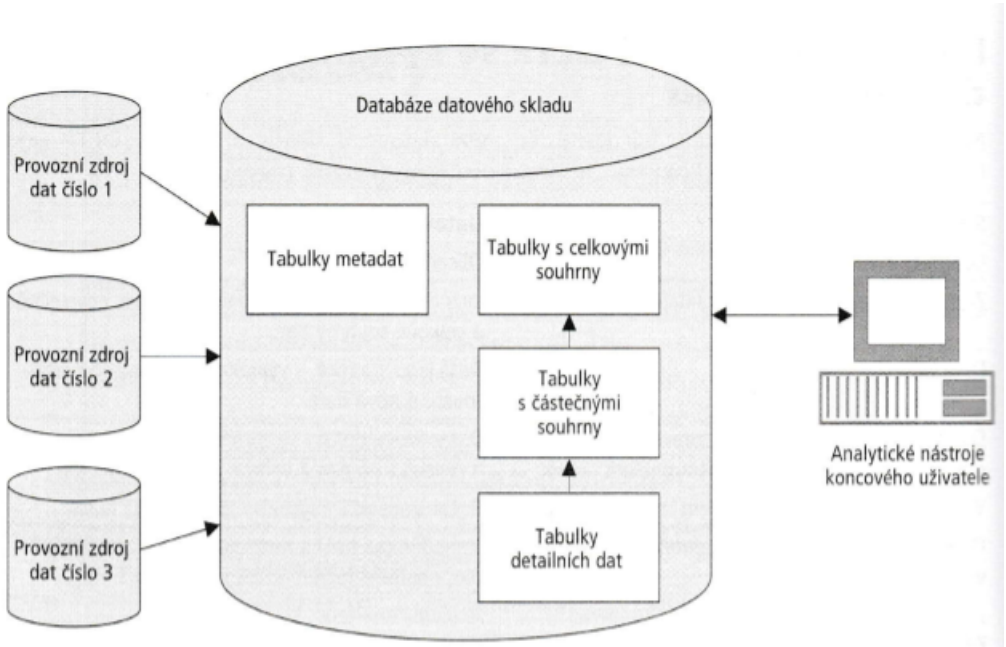
6.3 Architektura datových skladů

Architektura datových skladů se liší dle zvoleného přístupu, který byl popsán v kapitole o historii a vývoji datových skladů. Architekturu se přirovnává ke stavebním plánům. Vhodnou vizualizací zvolené architektury jsou diagramy toků, které v obecném pojetí popisují architekturu datových skladů jako jednoduchý projekt. Tento zdánlivě jednoduchý koncept z obrázku 6.1 - sběr zdrojových dat, centralizace do databáze a spuštění sestavy či reporting v nástroji BI vyžaduje velkou časovou kapacitu.

6.3.1 Architektura souhrnných tabulek

Zde hovoříme o architektuře dle Billa Inmona. Jeho pojetí silně porušuje princip normalizace. Nicméně v datovém skladu jsou uložena pouze historická data, která by se po zápisu neměla měnit. Historická data tečou z několik provozních zdrojů. Vhodně stanovená úroveň detailu a pochopení účelu značí úspěšné vytvoření datové skladu. Uživatel se dostává k podrobnějšímu rozboru (tzv. drill down).

Datový sklad se skládá z několik datových vrstev. Jádrem datových skladů jsou rozsáhlé databáze, které obsahují interní či externí data, která jsou navzájem integrovaná. Pod pojmem interní data si lze představit veškerá data získaná z provozních systémů v rámci společnosti v podobě každodenních transakcí. Tyto primární data udržuje spodní vrstva



Obrázek 6.1: Architektura datového skladu se souhrnnými tabulkami [34]

relační, na kterou navazuje vrstva dimenzionální.

Základem každého schématu datového skladu je faktová tabulka, která obsahuje atributy a měřítka. Faktová tabulka je provázaná s tabulkami dimeníz, které poskytují uživatelům pohlížet na data z různých pohledů. Dimenze nejsou vždy spojeny přímo s tabulkou faktů. Faktové tabulky a dimenze mohou tvořit několik schémat. Například schéma hvězda či sněhová vločka, které vyplývají z definici datového skladu dle Kimballa. [12].

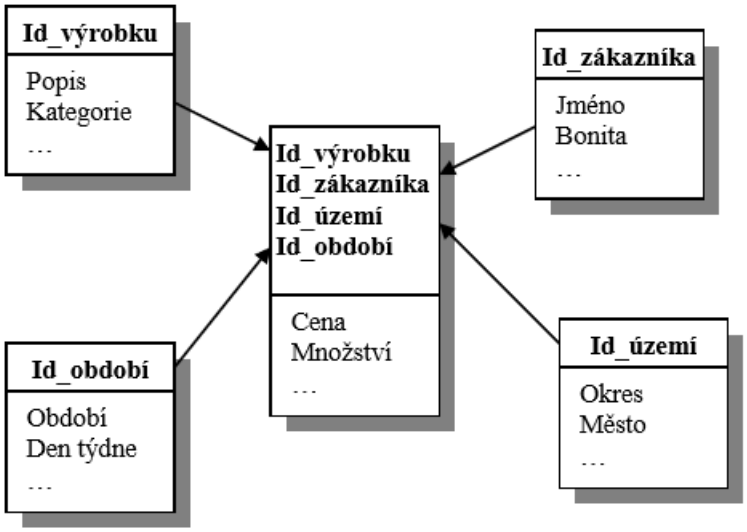
6.3.2 Multidimenzionální architektura

Multidimenzionální architektura představuje rozvržení prvků v rámci OLAP technologií. Příčinou vzniku této architektury je povinnost volby jiného modelu pro analýzy oproti modelu primárních dat. Základní proměnné dimenze jsou ukazatelé (ekonomické proměnné) a čas. Implementace tohoto přístupu probíhá ve dvou úrovních: Relační databáze (Star a Snowflake schéma) Speciální binární databáze

Schéma hvězda

Strukturu tvoří navzájem provázané dimenzní tabulky s tabulkou faktovou. Model na první pohled připomíná vzor hvězdy. Pokud bychom měli porovnávat tento model s modelem souhrnných tabulek, vidíme veliký rozdíl v návrhu logiky dat.

Každá z dimenzních tabulek je s tabulkou faktů svázaná relací 1:n a jsou navzájem propojeny pomocí referenční integrity. Hlavní výhodou tohoto modelu je snadné pochopení datového modelu a jednotlivých vazeb. Navržení hvězdicového schématu vyžaduje výbornou znalost oblast využití. Různá oddělení v rámci podniku budou mít rozdílné hvězdicové schémata.



Obrázek 6.2: Hvězdicové schéma[33]

Schéma sněhová vločka

Schéma sněhové vločky si lze představovat jako dimenze schématu hvězdy, které jsou propojené na ostatní dimenze. Schéma vločky dosáhneme normalizací jednotlivých dimenzí hvězdicového schématu. Model je vhodný při použití rozkouskování jednotlivých oddělení na divize či rozpad časového pohledu.

6.4 Druhy datových skladů

Tradiční datový sklad

Tradiční datový sklad se skládá z jednotlivých dávek dat, které obsahují platné časové razítko. V praxi se nastavují měsíční, čtvrtletní či roční dávky. Výsledné reporty se nesestavují dynamicky a ani ad-hoc, ale vždy ve stanoveném termínu, ke kterému tradiční datový sklad obsahuj aktuální data z produkce.

Near real-time datový sklad

Near real-time datový sklad je založen na přírůstkových dávkách dat. Oproti tradičnímu datovému skladu nedochází k aktualizaci všech částí, ale pouze k těm, které jsou aktualizované. Tím se tak stává účinnější a v určitém ohledu i rychlejší volbou.

Real-time datový sklad

Real-time datový sklad ukládá data v reálném čase. Tím jsou tak k analytickým dotazům k dispozici aktuálně platná data a lze tak rychle reagovat na business potřeby. Nevýhodou je značná náročnost na operace transakcí a tím se tak systém může zpomalit. A dochází tak v některých případech k zbytečné aktualizaci datového skladu, které není relevantní pro požadovaná data. Požadavky na analýzu aktuálních dat v reálném čase jsou hlavním hnacím motorem pro vznik real-time datových skladů.

Postupně se opouští od tradičního nočního dávkování. Cílem je efektivní nastavení dávkování. Efektivní i rozdělení nastavení procesu real-time datového skladu pro určitou oblast, například oddělení prodeje vyžaduje data real-time, zatímco finanční oddělení se spokojí s týdenní či měsíční dávkou dat.[35].

Jeden z produktů společnosti Oracle - ODI umožňuje spolu s Oracle GoldenGate integraci v reálném čase. Zachycení dat ze zdroje se provádí buď pomocí přírůstkových dotazů na základě časové značky či prostřednictvím mechanismu změny datového zachycení, který detekuje jakoukoliv změnu.

7 Praktická část

7.1 Definice problému

Datové sklady jsou tradiční doménou relačních databází, ale se zvyšující se popularitou NoSQL databází vzniká možnost jejich využití i v této oblasti. Častým použitím NoSQL databází je pro nestrukturovaná data, pro které není datové schéma jednoduše definované. Proto je žádoucí pro snadnější manipulaci používání aplikací Business Intelligence.

S tím ale vznikají problémy, jestli mají NoSQL databáze schopnost zastat funkci datového zdroje pro datový sklad. V SQL databázích je tvorba dotazů pomocí dotazovacího jazyka snadné pro získávání dat i pro netechnické uživatele. Prostřednictvím dotazů se získávají všechny data z databáze. To se tak stává nutným předpokladem pro jednoduchou integraci do datového skladu.

I když u datových skladů nebývá kladen velký důraz na rychlost dotazů, i přesto nastává situace, kde rychlý výsledek s aktuálními daty může být požadován. Je nutné se proto zaměřit i na to, zda-li NoSql obstojí pro definované kritérium. Jelikož často obsahují nestruktoavaná data, která představují množinu dat s neznámou strukturou a tím tak nastávají problémy.

Jedním z problémů je tzv. Full table scan. Jedná se o jednu z nejpomalejších operací dat v databázi, kde rychlost je přímo úměrná počtu záznamů. Jedná se o sken celé tabulky, kde je každý řádek načítán a data nejsou ničím omezena. Výsledkem jsou zajímavá data z celé tabulky a nikoliv jen z určité části.

Proto aby bylo možné zjistit praktickou využitelnost NoSql databází bude nutné vytvořit ukázkovou NoSql databázi a SQL databázi a použít je jako datový sklad. A rozdělit

oblast zájmu do tří oblastí:

Využitelnost agregačních a analytických dotazů

Sestavením agregačních dotazů a analytických dotazů v SQL a NoSQL bude měřena rychlost odezvy dotazu na základní množině dat.

Vliv růstu objemu dat na rychlost dotazování

Základní množina se multiplikuje a poroste tak objem dat. Měřením rychlosti agregačních a analytických dotazů ověříme časovou náročnost na zpracování dotazu.

Napojení NoSQL databáze pomocí konektoru na SQL databází

Pokud dojde k napojení NoSQL databáze na SQL a dotazování bude probíhat pouze v SQL jazyce, bude porovnána časová náročnost zpracování dotazu mezi tímto konektorem a NoSQL databází.

Ze zkoumání problematiky v praktické části očekáváme diskuzi nad výsledky a shrnutí výsledků v závěru včetně doporučeného budoucího směru zkoumání.

8 Praktická část - využitelnost agregačních funkcí a spojování tabulek

8.1 Volba technologií

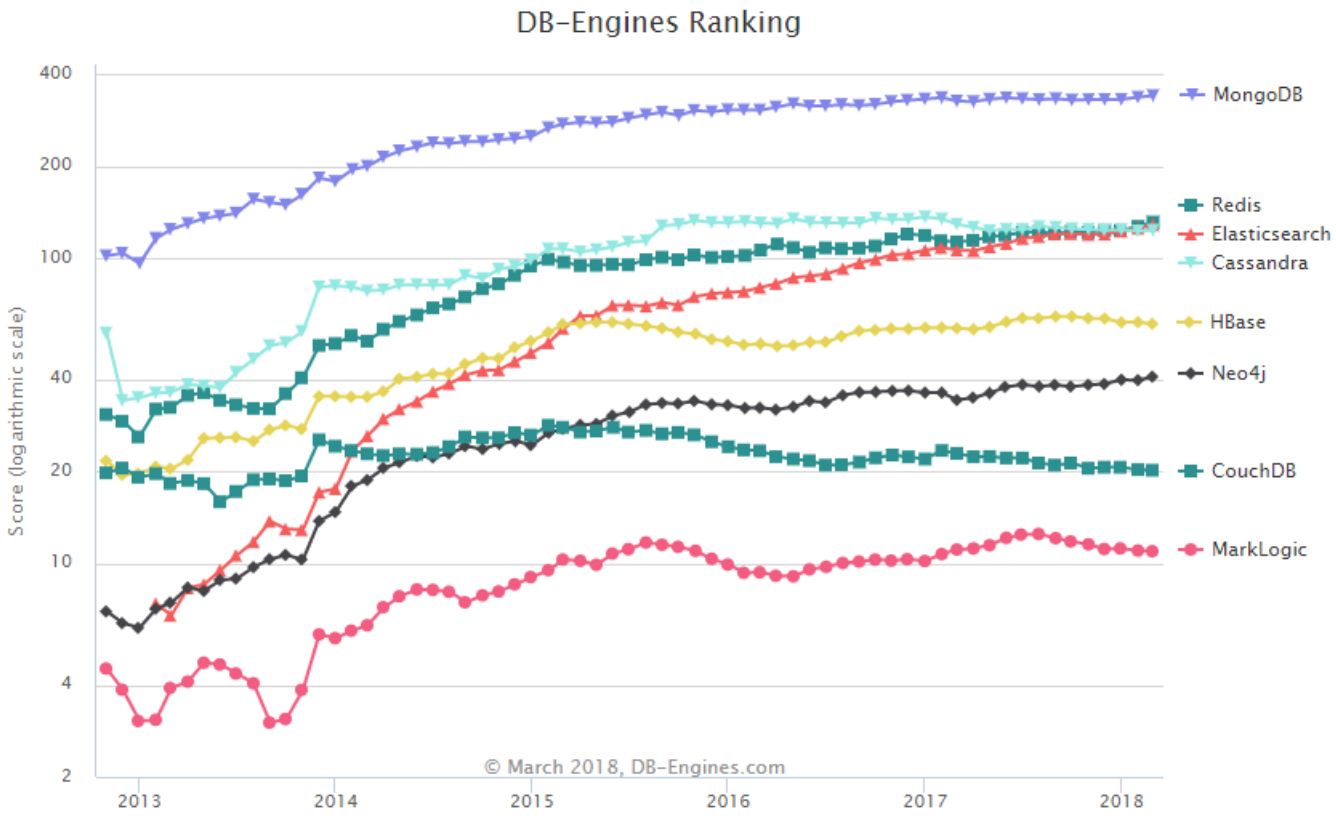
První databázový model je relační model tvořený v PostgreSQL. Důvodem zvolení PostgreSQL je open-source verze a dobrá interakce s ostatními aplikacemi pro dolování dat a reporting. Umožňuje analýzu, transformaci a poskytování modelovaných dat v rámci databázového serveru [36].

Druhý databázový model je nerelační tvořený v dokumentové databázi MongoDB. Důvodem zvolení tohoto nástroje je jeho široká rozšířenost (dle obrázku 8.1) a tvorba dynamických dotazů, tím se tak stává univerzálním NoSQL řešením. MongoDB ukládá data v binárním formátu JSON.

Veškeré databázové operace byly spuštěny na zařízení MacBook Pro (15-inch, 2017) s parametry:

- Rychlost procesoru: 2,8 GHz
- Název procesoru: Intel Core i7
- 16 GB 2133 MHz LPDDR3
- Počet procesorů: 1

- Celkový počet jader: 4
- Název zařízení: APPLE SSD SM0256L



Obrázek 8.1: Popularita NoSQL databází[37]

8.1.1 Datový model

V praxi získávají internetové obchody data z různých systémů a pak následně řeší integraci a tvorbu datového skladu. V diplomové práci je každá z použitých entit brána jako schéma, které je již složené z tabulek faktů. Jedná se tedy o zjednodušený ilustrativní model datového skladu internetového obchodu, jelikož cílem diplomové práce není popis tvorby samotného datového skladu, nýbrž využitelnost NoSQL v analytickém dotazování.

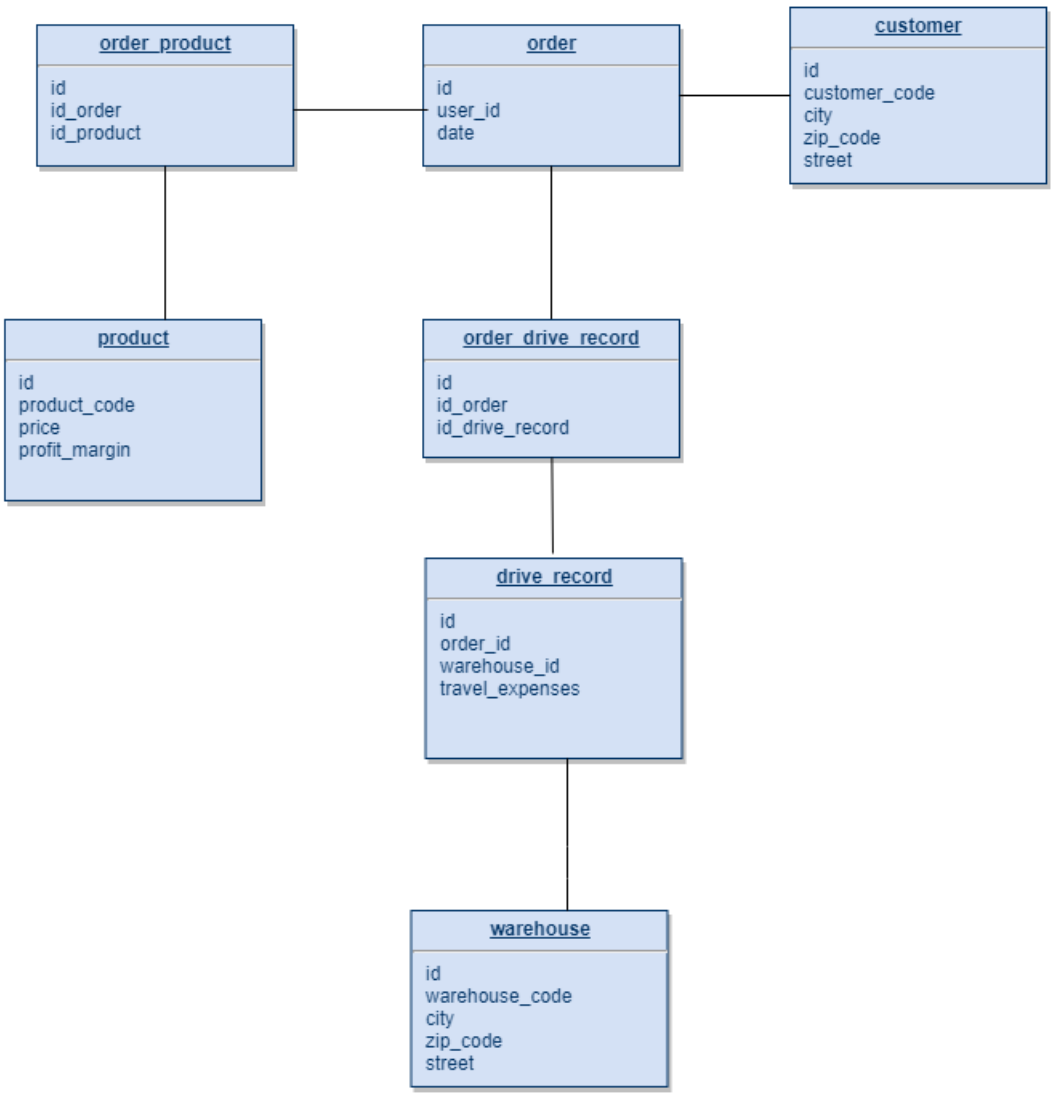
Datový model SQL

V SQL datovém modelu na obrázku 8.2 je reprezentován model internetového obchodu, který umožňuje pouze dopravu objednávek na adresu zákazníka, bez možnosti osobního vyzvednutí. V obchodě se evidují informace o objednávkách (`order`), zákaznících (`customer`), produktech (`product`), pobočkách (`warehouse`) a jízdách (`drive_record`) k jednotlivým zákazníkům. Každá tabulka má svůj jedinečný identifikátor. Mezi tabulkami se nachází vazební tabulka (`order_product`), která spojuje objednávky a produkty. Vazební tabulka (`order_drive_record`) spojuje tabulku pobočky a objednávky.

Jeden zákazník může vytvořit několik objednávek, nebo žádnou. Jedna objednávka vždy patří jednomu zákazníkovi. Objednávka vždy obsahuje jeden nebo více produktů, které jsou dopraveny na konkrétní adresu zákazníka z nejbližší pobočky.

Datový model NoSQL

Datový model NoSQL je postavený na stejných datech jako v SQL. V NoSQL nemáme předem definované schéma, ale rozdíly jsou v definování primárních klíčů, které v případě NoSQL nepředstavují číselné hodnoty, jelikož jsou generované dle tabulky a čísla záznamu. Z důvodu jednodušší práce s generátorem dat. Vazby 1:N jsou řešeny pomocí one-to-few (pro tabulky `drive_record` a `order`) a one-to-many (tabulky `order` a `product`) dle kapitoly 5.3.3 v teoretické části.



Obrázek 8.2: SQL databázový model nakreslený v draw.io

8.2 Příprava dat

Diplomová práce využívá uměle vygenerovaná data pro SQL a NoSQL. Generátor dat byl vytvořen v JavaScriptu. Využívá knihovnu Faker pro náhodná data a knihovnu SeedRandom z důvodu opakovatelnosti generovaných výsledků. Pro připojení do databází jsou využity oficiální knihovny - PG a MongoDB. Data jsou pouze ilustrační a nemají žádnou vypovídající hodnotu. Zdrojový kód generátoru je uveden v příloze.

Základní množství záznamů je následující:

- Počet adres - 10
- Počet uživatelů - 1000
- Počet produktů - 10000
- Počet skladů - 10
- Počet objednávek - 5000. Každá objednávka obsahuje 1 - 50 záznamů produktů.
- Počet záznamů jízdy - 500. Pro každý záznam jsou přiřazeny objednávky pro dané datum.

Uvedené množství dat je možné vynásobit multiplikátorem a získat tak větší objem dat ve stejném poměru. Ukázku kódu vkládání dat lze spatřit na obrázku 8.3

```
// console.log("nosql - creating driver records");
for (let i = 0; i < orderDates.length; i++) {
  const date = orderDates[i];
  const orders = await db
    .collection("order")
    .find({ date })
    .toArray();
  const warehouseId = randomIntFromInterval(0, warehousesCount);
  await db.collection("drive_record").insertOne({
    _id: "drive_record" + i,
    travel_expenses: randomIntFromInterval(10, 500),
    warehouse: "warehouse" + warehouseId,
    orders
  });
}
```

Obrázek 8.3: Vkládání dat pro kolekci `drive_record`

8.3 Testování

Testování bylo prováděno pomocí dvouvýběrového nepárového T-testu s různým rozptylem a hladinou spolehlivosti 0,05 nad základní množinou dat s hypotézou:

- Čas běhu NoSQL DB je menší než SQL DB

Výpočet T-testu byl proveden pomocí funkce ttest v Javascriptu a ke každé agregační funkci byly zjištěny tyto hodnoty intervalu spolehlivosti:

- dolní interval - dolní hranice intervalu spolehlivosti (horní hranice je nekonečno)
- p value - kritický obor
- t value - testové kritérium

  Všechny výsledky studentova T-testu byly naměřeny v nanosekundách a zaokrouhleny na dvě desetinná místa, jelikož se při měření často objevovaly hodnoty 1 a 0, které by neměly žádnou vypovídající hodnotu.

8.4 Agregační funkce v SQL vs NoSQL

Nad vygenerovanými daty byly provedeny základní agregační dotazy. Agregační dotaz byl spuštěn pouze nad jednou tabulkou. Pro sestavení agregačních dotazů v NoSQL byl použit princp agregačních pipelines, které popisuje kapitola 5.3.5 Bylo provedeno porovnání rychlosti výsledného dotazu a na první pohled jsou zřejmé i rozdíly v psané syntaxi dotazu. Rozhodnout jaký dotaz je pro psaní vhodnější je otázkou osobní preference a zvyku.

Pro zjištění rychlosti agregačních dotazů byl vytvořen jednoduchý skript v JvaScriptu, který porovnává rychlost v SQL a v NoSQL. Každá z funkcí byla spuštěna 100 krát.

COUNT

Dotaz pro zjištění počtu všech produktů v databázi. Podobu dotazu v SQL lze spatřit na obrázku 8.4 Pro sčítací funkci byl v NoSQL použit zkrácený tvar psaného dotazu, viz ukázka 8.5 Výsledky studentova testu jsou v tabulce 8.1 a 8.2.

```
select count(price) from product
```

Obrázek 8.4: dotaz COUNT v SQL

```
product.countDocuments();
```

Obrázek 8.5: dotaz COUNT v NoSQL

Agregační funkce	MIN	MAX	MEDIAN	AVG
COUNT SQL	0,76	4,14	0,82	0,87
COUNT NoSQL	2,96	5,43	3,05	3,17

Tabulka 8.1: Výsledky zpracování dotazu COUNT

Agregační funkce	Dolní interval	p value	t value	Potvrzena hypotéza
COUNT	2138669.59	0	28,06	NE

Tabulka 8.2: Výsledky T-testu zpracování dotazu COUNT

SUM

Dotaz provede operaci sčítání pro všechny ceny produktů.V SQL vypadá dotaz dle obrázku 8.6 a v NoSQL je syntaxe dle obrázku 8.7. Výsledky jsou v tabulce 8.3 a 8.4.

```
select sum(price) from product
```

Obrázek 8.6: dotaz SUM v SQL

```
product
  .aggregate([{
    $group: {
      _id: null,
      total: {
        $sum: "$price"
      }
    }
  }])
```

Obrázek 8.7: dotaz SUM v NoSQL

Agregační funkce	MIN	MAX	MEDIAN	AVG
SUM SQL	0,78	1,72	0,84	0,89
SUM NoSQL	7,29	9,3	7,67	7,77

Tabulka 8.3: Výsledky zpracování dotazu SUM

Agregační funkce	Dolní interval	p value	t value	Potvrzena hypotéza
SUM	7101207.82	0	132.58	NE

Tabulka 8.4: Výsledky T-testu zpracování dotazu SUM

AVG

Dotaz pro operaci average zjistí průměrnou cenu produktu, z celkového seznamu produktů. Dotaz v SQL lze spatřit na obrázku 8.8 a v NoSQL na obrázku 8.9. Výsledky testu jsou v tabulce 8.5 a 8.6.

```
select avg(price) from product
```

Obrázek 8.8: Dotaz AVG v SQL

```
.aggregate([{
  $group: {
    _id: null,
    total: {
      $avg: "$price"
    }
  }
}])
```

Obrázek 8.9: Dotaz AVG v NoSQL

Agregační funkce	MIN	MAX	MEDIAN	AVG
AVG SQL	0,78	1,58	0,85	0,88
AVG NoSQL	6,99	12,04	7,56	7,94

Tabulka 8.5: Výsledky zpracování dotazu AVG

Agregační funkce	Dolní interval	p value	t value	Potvrzena hypotéza
AVG	6786078.27	0	28,06	NE

Tabulka 8.6: Výsledky T-testu zpracování dotazu AVG

MIN

Dotaz pro zjištění minimální ceny produktu. Dotazy lze spatřit na obrázku 8.10 a 8.11 Výsledky testu jsou v tabulce 8.7 a 8.8.

```
select min(price) from product
```

Obrázek 8.10: Dotaz MIN v SQL

```
product
  .aggregate([{
    $group: {
      _id: null,
      total: {
        $min: "$price"
      }
    }
  }])
```

Obrázek 8.11: Dotaz MIN v NoSQL

Agregační funkce	MIN	MAX	MEDIAN	AVG
MIN SQL	0,79	1,69	0,85	0,89
MIN NoSQL	6,93	9,04	7,5	7,65

Tabulka 8.7: Výsledky zpracování dotazu MIN

Agregační funkce	Dolní interval	p value	t value	Potvrzena hypotéza
MIN	6679923.19	0	142.89	NE

Tabulka 8.8: Výsledky T-testu zpracování dotazu MIN

MAX

Dotaz pro zjištění maximální ceny produktu. Dotazy lze spatřit na obrázku 8.12 a 8.13 Výsledky testu jsou v tabulce 8.9 a 8.10.

```
select max(price) from product")
```

Obrázek 8.12: Dotaz MAX v SQL

```
product
    .aggregate([{
      $group: {
        _id: null,
        total: {
          $max: "$price"
        }
      }
    }])
```

Obrázek 8.13: Dotaz MAX v NoSQL

Agregační funkce	MIN	MAX	MEDIAN	AVG
MAX SQL	0,79	1,58	0,90	0,94
MAX NoSQL	7,02	8,89	7,49	7,56

Tabulka 8.9: Výsledky zpracování dotazu MAX

Agregační funkce	Dolní interval	p value	t value	Potvrzena hypotéza
MAX	6566220.06	0	158.96	NE

Tabulka 8.10: Výsledky T-testu zpracování dotazu MAX v ms

8.5 Spojování tabulek one-to-few v SQL vs NoSQL

V této části diplomové práce byl použit princip one-to-few, pomocí kterého jsme dosáhli spojení tabulek a vyřešní vazby 1:N. Pro vyhodnocení komplexnosti a rychlosti dotazování je stanoven tento analytický požadavek:

- Manažer obchodu si přeje zobrazit seznam všech unikátních produktů k dané objednávce s vazbou na záznam jízdy

Testování bylo prováděno nad základním množstvím dat.

8.5.1 Řešení v SQL

Pro nalezení unikátnosti výsledku byl použit DISTINCT hodnot identifikátorů produktu. V SQL dotazu došlo ke spojení dvou tabulek za pomocí JOIN. Ukázku příkazu lze spatřit na obrázku 8.14.

```
SELECT DISTINCT po.id_product FROM drive_record dr
    JOIN order_drive_record odr ON (dr.id = odr.id_drive_record)
    JOIN order_product po on (odr.id = po.id_order)
    WHERE dr.id = 2`
```

Obrázek 8.14: Dotaz v SQL

8.5.2 Řešení v NoSQL

Z důvodu vazby one-to-few není zapotřebí spojování tabulek, jelikož tabulka již obsahuje daná data. Proto je možné nad nimi přímo pracovat. Pro omezení výsledku byl použit operátor MATCH a pro unikátnost výsledku nemá MongoDB konkrétní operátor, ale je zapotřebí využít operátor reduce a v něm použít funkci setUNION, jež je obrazem matematického sjednocení množin. Ukázku příkazu lze spatřit na obrázku 8.15

```
driveRecord
  .aggregate([
    { $match: { _id: "drive_record2" } },
    {
      $project: {
        products: {
          $reduce: {
            input: "$orders.products",
            initialValue: [],
            in: { $setUnion: ["$$value", "$$this"] }
          }
        }
      }
    }
  ])
```

Obrázek 8.15: One-to-few v NoSQL

8.5.3 Výsledky

Pro zjištění rychlosti dotazu se spojením více tabulek byl vytvořen jednoduchý skript v JavaScriptu, který porovnává rychlost v SQL a v NoSQL. Každý dotaz byl spuštěn 100 krát a jako ochrana proti náhodným zpomalením byl výsledek naměřených hodnot zprůměrován.

Pro ilustraci, jak by se dotaz choval bez omezení na daný výsledek, byl spuštěn bez WHERE a match podmínky. Výsledky testu jsou v tabulce 8.11 a 8.12.

Příkaz	MIN	MAX	MEDIAN	AVG
SQL	4,27	10,59	4,43	4,65
NoSQL one-to-few s where podmínkou	0,44	1,67	0,49	0,58
NoSQL one-to-few bez match a where podmínky	71,18	101,2	75,68	78,37

Tabulka 8.11: Výsledky zpracování dotazu one-to-few v ms

Příkaz	Dolní interval	p value	t value	Potvrzena hypotéza
one-to-few s where podmínkou	1801048.76	0	51.61	NE
one-to-few bez match a where podmínky	48057403.87	0	177,30	NE

Tabulka 8.12: Výsledky T-testu zpracování dotazu one-to-few v ms

8.6 Spojování tabulek one-to-many v SQL vs NoSQL

Pro komplexnější analytické dotazování přes větší množství entit se používá pro spojení tabulek v SQL JOIN klauzule, nebo za pomocí where podmínky. Spojení tak umožňuje formulovat vztah 1:N. V NoSQL je používán jiný přístup tvorby dotazu a i jiná sekvence úkonů, které se v rámci dotazování provádějí. V této části diplomové práci byl použit princip one-to-many. Pro vyhodnocení komplexnosti a rychlosti dotazování je stanoven tento analytický požadavek:

- Manažer obchodu si přeje zobrazit seznam všech produktů k dané objednávce s omezením na určitý den

Řešení v SQL

Pořadí zpracování dotazu v SQL (viz obrázek 8.16 je rozdílné, oproti napsanému dotazu. Jako první dochází k výběru množiny objednávek. Dále se provede spojení tabulek přes klíčové identifikátory. Tato celková množina se zpracovává omezením v podmínce WHERE. Výběr z množiny je omezen definicí v klauzuli SELECT. Na závěr dochází k seřazení dle identifikátoru objednávky.

```
SELECT o.id, o.date, pr.product_code
  FROM public.order o
  JOIN order_product op ON (op.id_order = o.id)
  JOIN product pr ON (pr.id = op.id_product)
  WHERE o.date = '2019-02-21'
  ORDER BY o.id
```

Obrázek 8.16: One-to-many v SQL

Řešení v NoSQL

Vazba one-to-many je pro NoSQL komplikovanější a řeší se pomocí operátoru LOOKUP. Operátor nelze v tomto případě přímo použít, ale musí mu předcházet další kroky. Celý dotaz je vysvětlen následovně: Pro sestavení NoSQL dotazu na obrázku 8.17 byly dodrženy následující kroky:

1. Match operátor na omezení výsledku
2. Příprava pro spojení

   Jelikož je seznam produktů v dané objednávce realizován pomocí pole a s tím neumí LOOKUP přímo operovat. Je nutné toto pole rozvinout tak, aby každý záznam v poli byl samostatným záznamem v celkové kolekci. K tomu je použit operátor UNWIND. Ve výsledku tedy z jednoho záznamu objednávky o deseti produktech dostaneme deset těch samých záznamů objednávky, ale pouze s konkrétním produktem.
3. Pro vytvoření spojení tabulek se používá operátor LOOKUP, který se skládá ze čtyř parametrů:
   a) FROM - určení z jaké tabulky se vybírají data
   b) LOCAL FIELD - výsledky z operace UNWIND
   c) FOREIGN FIELD - unikátní identifikátor
   d) AS - alias výsledku operace LOOKUP

   Po provedení operace se provede spojení tabulek a výsledkem je pole dat.
4. Příprava pro projekci

   Vzhledem k tomu, že výsledkem je pole, je zapotřebí použít UNWIND, abychom dostali konkrétní záznam.
5. Projekce dat

   Konkrétní výběr sloupců v klauzuli select v SQL se v NoSQL řeší operací PROJECT, která se skládá z definujících parametrů pro konkrétní výběr. Dle příkladu pracujeme s identifikátorem objednávky, datu objednávky a produktového kódu.

```
order
  .aggregate([
    { $match: { date: new Date("2018-11-13T02:43:20.103Z") } },
    {
      $unwind: "$products"
    },
    {
      $lookup: {
        from: "product",
        localField: "products",
        foreignField: "_id",
        as: "joined_product"
      }
    },
    { $unwind: "$joined_product" },
    {
      $project: {
        _id: 1,
        date: 1,
        product_code: "$joined_product._id"
      }
    }
  ])
```

Obrázek 8.17: One-to-many v NoSQL

Výsledky

Testování probíhalo stejným způsobem jako u předchozího příkladu a výsledky testu jsou v tabulce 8.13 a 8.14.

Příkaz	MIN	MAX	MEDIAN	AVG
SQL	0,53	2,05	0,55	0,6
one-to-many s where podmínkou	2,24	3,48	2,4	2,53
one-to-many bez where podmínky	2440,26	5033,92	2610,91	2659,91

Tabulka 8.13: Výsledky zpracování dotazu se spojením více tabulek v ms

Příkaz	Dolní interval	p value	t value	Potvrzena hypotéza
one-to-many s where podmínkou	- 419140	1	- 55,68	ANO
one-to-many bez where podmínky	2338604775,95	0	493,14	NE

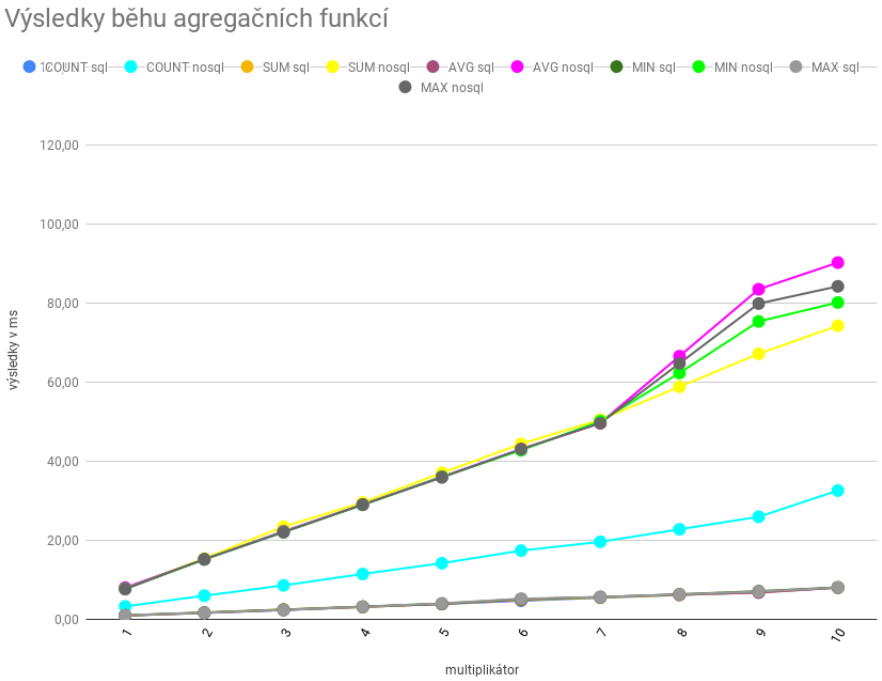
Tabulka 8.14: Výsledky T-testu zpracování dotazu se spojením více tabulek v ms

8.7 Vliv rozsahu dat na rychlost dotazu

Jelikož se v datových skladech pracuje s velkým množstvím dat, je zapotřebí zkoumat rychlost dotazu i v závislosti na množství dat v databázi. Proto byl vytvořen porovnávací skript, který v každém svém běhu zdvojnásobil množství dat oproti předchozímu běhu. Celkem bylo provedeno deset běhů, kde každý běh znásobuje základní set dat. Byly použity stejné agregační dotazy, jako v předchozí kapitole.

8.7.1 Agregační funkce

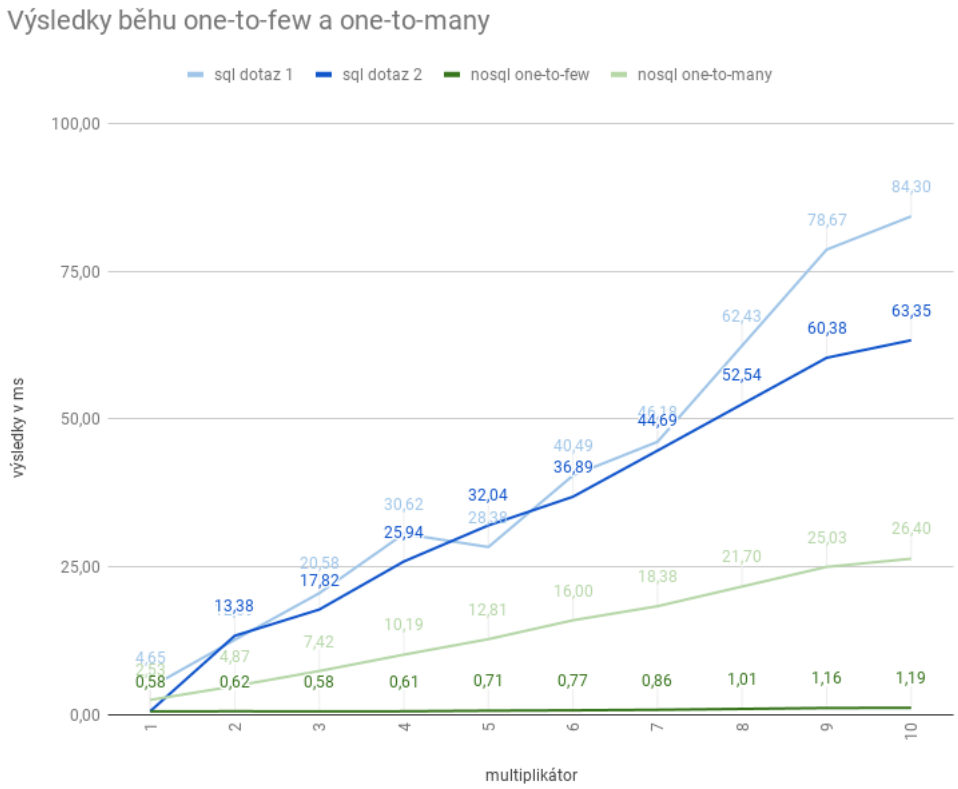
Na následujícím grafu 8.18 je vidět trend vývoje v databázích SQL a NoSQL pro agregační funkce. Bod pro každý běh představuje průměrnou hodnotu běhu dotazu, který byl spuštěn 100 krát. Z výsledků je patrné, že se rozdíl mezi SQL a NoSQL se s přibývajícím množstvím dat zvětšuje a není konstantní. NoSQL databáze je čím dál tím víc pomalejší, než SQL databáze. Pouze v případě funkce COUNT, je rozdíl konstatnější než u ostatních funkcí.



Obrázek 8.18: Výsledky běhu agregačních funkcí

8.7.2 JOIN

Na následujícím grafu 8.19 je vidět trend vývoje v databázích SQL a NoSQL pro spojení dvou tabulek principů one-to-many (v SQL dotaz 2) nebo one-to-few (v sql dotaz 1). Bod pro každý běh představuje průměrnou hodnotu běhu dotazu, který byl spuštěn 100 krát. Z výsledků je patrné, že se rozdíly mezi SQL a NoSQL s přibývajícím množstvím dat zvětšují. Zatímco při dotazu one-to-few se NoSQL databáze téměř nezvyšuje čas potřebný k vykonání dotazu u SQL roste v závislosti na počtu dat. A i v případě druhého dotazu pro one-to-many je tento rozdíl patrný.



Obrázek 8.19: Výsledky běhu one-to-few a one-to-many

9 Využitelnost NoSQL ve stávajících systémech Datových skladů

V případech existence NoSQL databáze je řešením pro rozsáhlé agregační dotazy konektory převodu NoSQL databáze na SQL. Konektory jsou známé pod zkratkou ODBC, která představuje standardizované softwarové API pro přístup k databázovým systémům s dotazováním pro operace nad daty s SQL.

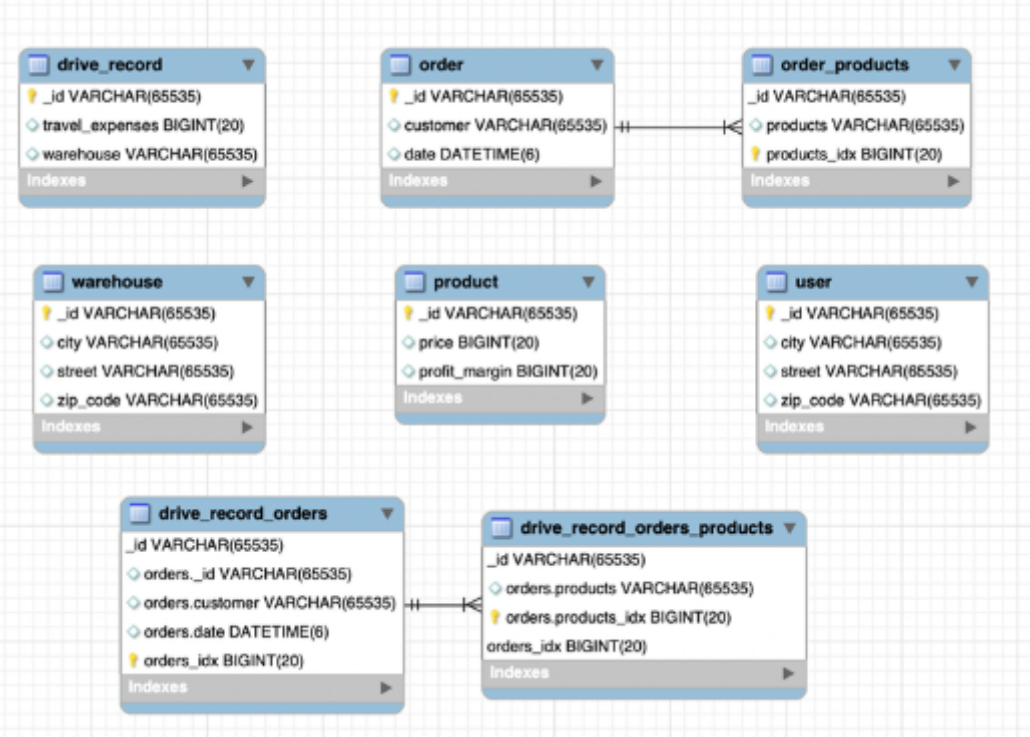
V této sekci praktické části byla provedena analýza konektoru MongoDB, rozbor tvorby dotazů a samotné měření rychlosti stejných dotazů nad stejnými daty, jako v kapitole 8.4 a 8.6

9.1 Konektor MongoDB

V diplomové práci byl použit driver pro MongoDB zvaný MongoDB Connector, který pracuje nad stejnými daty jako v kapitole 8.2. Díky tomuto nástroji je snadné se připojit k MongoDB a provádět analytické operace nad daty. Konektor poskytuje přístup k datům abstrahováním hierarchické struktury dokumentů a reprezentaci dat v podobě tabulek přes rozhraní MySQL. Tento konektor funguje jako proxy server SQL a převádí příchozí dotazy v SQL na ekvivalentní jazyk dotazu v MongoDB. Výsledky jsou vráceny v tabulkovém formátu a mohou být dále využity pro BI nástroje, pro které je zaručena podpora připojení k MySQL.

9.1.1 Práce s konektorem a DB schéma

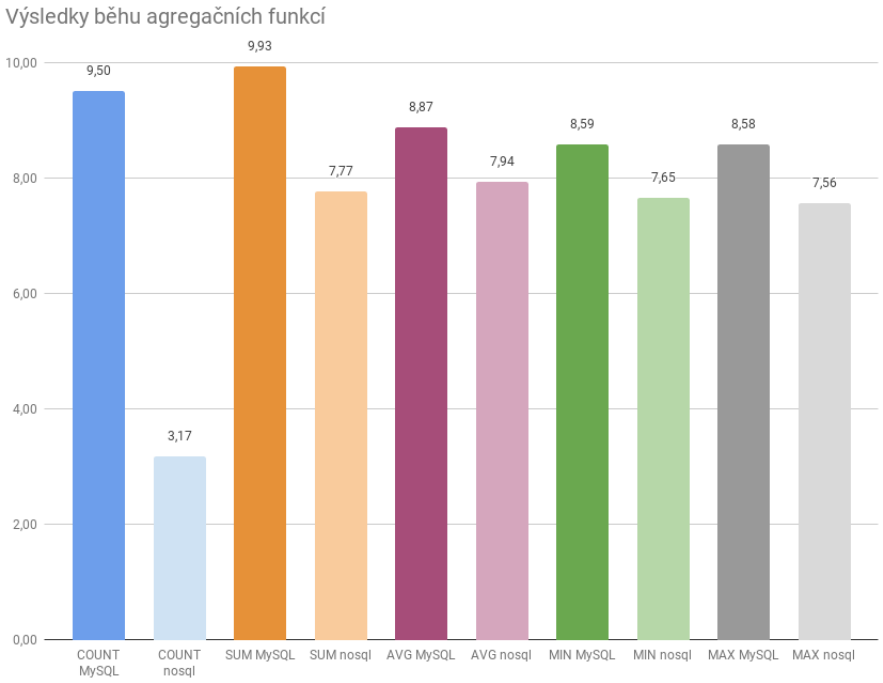
Instalace konektoru je jednoduchá, rychlá a volně přístupná. Po jeho napojení na MongoDB DB se z dat v NoSQL vygenerovalo schéma. Pokud bychom porovnali SQL schéma v Postgre na obrázku 8.2 z předchozí kapitoly praktické části a nově vygenerované schéma na obrázku 9.4, zjistíme, že si konektor navíc vygeneroval dvě nové vazební tabulky `drive_record_orders` a `drive_record_orders_products`.



Obrázek 9.1: DB schéma po transformaci z MySQL

9.2 Agregační funkce v konektoru vs NoSQL

V této kapitole bylo provedeno měření pro základní množinu dat. Byly spuštěny agregační funkce popsané v kapitole 8.4 . Byly porovnány časové rozdíly zpracování v konektoru versus spouštění dotazu přímo v NoSQL. Výsledky v milisekundách byly znázorněny na grafu 9.2. Pro vyhnutí se náhodnému zpomalení, byly data spuštěny 100 krát a výsledek byl zprůměrován.

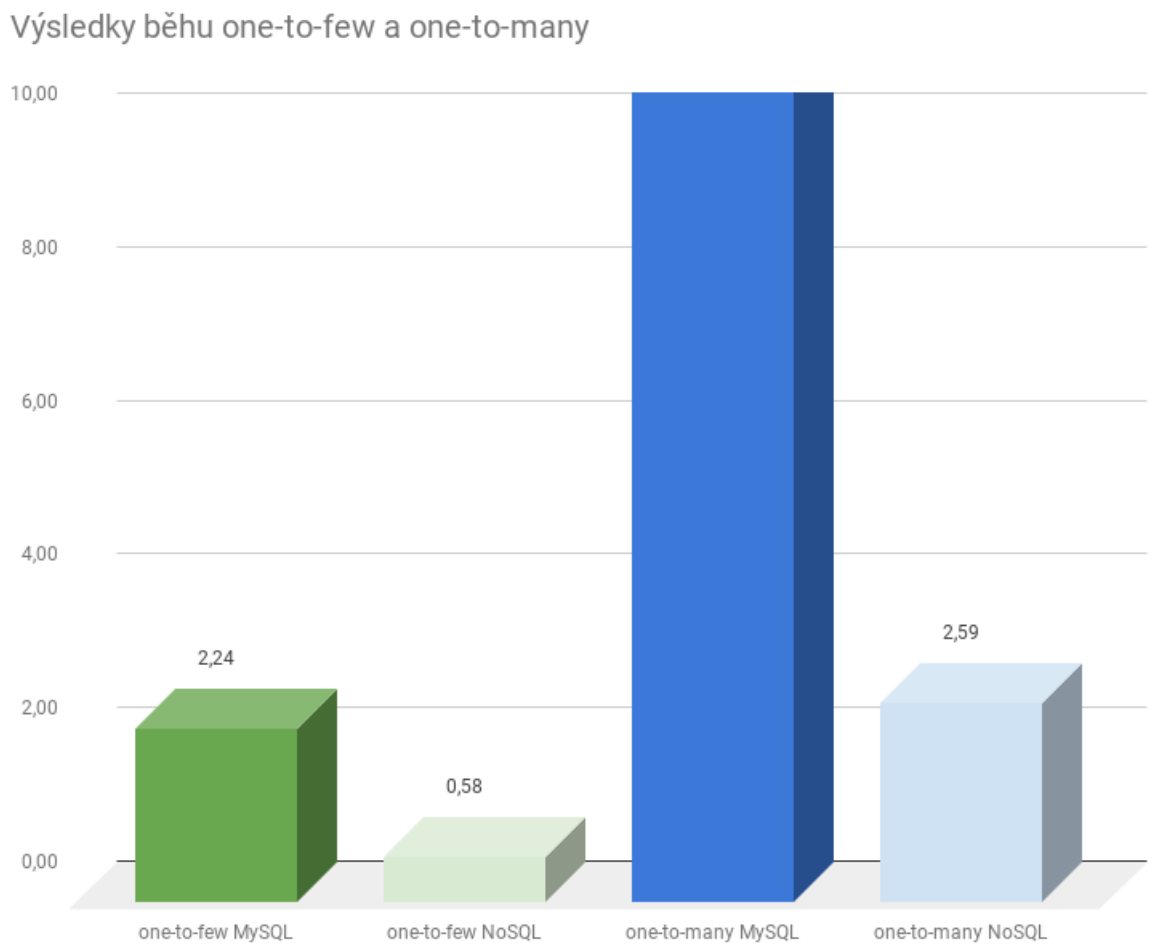


Obrázek 9.2: Výsledky běhu agregačních funkcí

Jak je z grafu patrné, využitím konektoru se čas pro vrácení dotazu zpomalí. Zpomalený výsledek u funkce COUNT byl byl očekávaný, jelikož samotná transformace z SQL dotazu na NoSQL není snadnou operací.

9.3 Spojování tabulek v konektoru vs NoSQL

V této kapitole bylo provedeno měření na základní množině dat pro dotazy, které zaručují spojení tabulek, podobně, jak je tomu v kapitole 8.5. Dochází zde k porovnání spuštění dotazu prostřednictvím konektoru napojeného na MySQL a dotazy spuštěné přímo z NoSQL databáze. Jako v předchozím měření očekáváme mírné zpomalení systému, jelikož zde dochází k transformaci SQL dotazu na NoSQL. Výsledky jsou zobrazeny v milisekundách na obrázku 9.3. Pro vyhnutí se náhodnému zpomalení, byly data spuštěny 100 krát a výsledek byl zprůměrován.



Obrázek 9.3: Výsledky běhu one-to-few a one-to-many

Princip tvorby dotazů

Jelikož výsledky při generování dat pro spojené tabulky dle principu one-to-many z konektoru byly oproti generováním z MongoDB několikanásobně vyšší, byl proveden rozbor generovaného dotazu. Pokud se podíváme na dotaz provedený přímo v MongoDB dle obrázku 8.15 a na dotaz generovaný z logu v databázi konektorem dle obrázku 9.4 zaznamenáme na první pohled rozdíly v obsáhlosti dotazů.


```
{
    "op": "command",
    "ns": "dipl.order",
    "command": {
      "aggregate": "order",
      "pipeline": [
        {
          "$unwind": {
            "path": "$products",
            "includeArrayIndex": "products_idx",
            "preserveNullAndEmptyArrays": false
          }
        },
        {
          "$match": {
            "products": {
              "$ne": null
            }
          }
        },
        {
          "$lookup": {
            "as": "__joined_pr",
            "from": "product",
            "localField": "products",
            "foreignField": "_id"
          }
        },
        {
          "$unwind": {
            "path": "$__joined_pr",
            "preserveNullAndEmptyArrays": false
          }
        },
        {
          "$match": {
            "_id": {
              "$ne": null
            }
          }
        },
        {
          "$lookup": {
            "foreignField": "_id",
            "as": "__joined_o",
            "from": "order",
            "localField": "_id"
          }
        },
        {
          "$unwind": {
            "preserveNullAndEmptyArrays": false,
            "path": "$__joined_o"
          }
        },
        {
          "$match": {
            "$and": [
              {
                "__joined_o.date": {
                  "$gte": "ISODate(2019-02-21T00:00:00Z)"
                }
              },
              {
                "__joined_o.date": {
                  "$lt": "ISODate(2019-02-22T00:00:00Z)"
                }
              }
            ]
          }
        },
        {
          "$sort": {
            "__joined_o._id": 1
          }
        },
        {
          "$limit": "NumberLong(50000)"
        },
        {
          "$project": {
            "dipl_DOT_o_DOT__id": "$__joined_o._id",
            "dipl_DOT_o_DOT_date": "$__joined_o.date",
            "dipl_DOT_pr_DOT__id": "$__joined_pr._id",
            "_id": 0
          }
        }
      ],
      "$db": "dipl"
    }
}
```


Obrázek 9.4: Kód z logu databáze MySQL

10 Diskuze nad výsledky

Při tvorbě ukázkového datového skladu v NoSql nebyly nalezeny žádné markantní rozdíly o proti SQL. V návrhu modelu bylo zjištěno, že při zpracování velkého množství dat se inklinuje v NoSQL k podobným návrhům řešení vazby 1:N, jak je tomu v SQL. Jelikož MongoDB má omezení na maximální velikost dokumentu, která nesmí přesáhnout 16 MB a tak není možné denormalizovat veškerá data. Při návrhu modelu bylo toto zohledňováno.

Pomocí agregačních pipeline byly v MongoDB realizované stejné operace, jako v případě SQL dotazovacího jazyka. Na rozdíl od SQL dotazů, je jejich komplexita a technická složitost daleko větší. Např. v případě unikátního záznamu, kdy v SQL lze použít DISTINCT, tak v NoSql je nutné mít znalost několika operátorů a i jejich vnitřního fungování. Naopak výhodná je možnost větší transformace dat přímo v dotazu a tím tak dochází k odlehčení zátěže v systému a neděje se proto transformace na úrovni aplikační logiky, jako je tomu v SQL řešení.

V testování agregačních funkcí přes celou tabulku, je zcela patrné, že SQL představuje vhodnější volbu a nepotvrdila se tak hypotéza, že NoSQL databáze je v tomto případě rychlejší. Nicméně naměřená rychlost pro NoSQL v praktické použitelnosti není negativní. I v případě zpracování tak velkého množství dat je celková operace časově únosná.

Složitější analytické funkce se spojováním tabulek v testováním prokázaly odlišné výsledky oproti agregačním funkcím.

Při využití výhody uložení přímo spojovaných dat do dané tabulky, rychlost dotazu je stále stejná oproti rostoucí rychlosti v SQL. V druhém případě složitější vazby, kde je nutné spojovat i NoSql tabulky, má MongoDB také navrch. Stále se jedná i pro SQL zcela únosnou mez vykonání dotazu.

Vzhledem k zřejmému rychlostnímu rozdílu v zcela základních dotazech nebylo

zapotřebí ukazovat složitější operace, jelikož by trend uchyloval k podobným výsledkům. Použité dotazy pokrývají celou množinu základních analytických dotazů.

11 Závěr

Cílem diplomové práce bylo nalezení využitelnosti NoSQL databází v datových skladech. Inspirace byla poskytnutá pomocí rešerše ostatních prací, které se zabývaly podobným tématem. V teoretické části byly představeny základní pojmy, které souvisí se zkoumanou problematikou. Díky analýze jsem byla schopna zvolit si vhodné nástroje pro testování a výzkum v této diplomové práci.

Využitelnost byla testována v praktické části z hlediska rychlosti agregačních dotazů a dotazů pro spojování více tabulek. Praktická část byla rozdělena na tři etapy. První etapa zkoumala dotazy na základní množině dat a ověřovala výsledky pomocí dvouvýběrového nepárového T-testu s různým rozptylem. Druhá etapa zkoumala vliv počtu dat na rychlost dotazů. S ohledem na delší čas generování byly použity hodnoty z průměru výběru, které byly graficky porovnány. Třetí etapa se zabývala konektorem, který napojil NoSQL databázi na SQL a pomocí SQL dotazů byla porovnávaná rychlost dotazu z konektoru a z NoSQL.

Pouze jeden výsledek potvrdil hypotézu, při které bylo dokázáno, že NoSQL databáze jsou rychlejší než SQL a to v případě spojení tabulek principem one-to-many. Tento předpoklad se potvrdil i při testu vlivu objemu dat na rychlost dotazu. Na základě těchto výsledků mohu konstatovat, že NoSQL databáze jsou vhodné pro druhy analytických dotazů, které vyžadují spojování tabulek. Ale při použití agregačních funkcí jsou vhodnější SQL řešení. Této situaci ani nepomůže konektor, který je napojen na NoSQL databázi a dotazujeme se pomocí SQL.

Budoucí vývoj práce může spočívat ve zkoumání NewSQL databází a porovnání mezi SQL a NoSQL.

Literatura

[1] (). The IDC data warehousing ROI study: An analysis, SearchSQLServer, WWW: http://searchsqlserver.techtarget.com/tip/The-IDC-data-warehousing-ROI-study-An-analysis (cit. 14. 01. 2018).

[2] B. J. Drnek, "Analýza NoSQL databází a jejich srovnání", s. 76,

[3] (). Výhody a nevýhody relačních a nerelačních (noSQL) databází pro analytické úlohy - VŠKP - VŠE, WWW: https://vskp.vse.cz/45052_vyhody_anevyhody_relacnich_anerelacnich_nosql_databazi_pro_analyticke_ulohy (cit. 11. 04. 2019).

[4] Z. Parker, S. Poe a S. V. Vrbsky, "Comparing NoSQL MongoDB to an SQL DB", in *Proceedings of the 51st ACM Southeast Conference on - ACMSE '13*, Savannah, Georgia: ACM Press, 2013, s. 1, ISBN: 978-1-4503-1901-0. DOI: 10.1145/2498328.2500047. WWW: http://dl.acm.org/citation.cfm?doid=2498328.2500047 (cit. 11. 04. 2019).

[5] Y.-T. Liao, J. Zhou, C.-H. Lu, S.-C. Chen, C.-H. Hsu, W. Chen, M.-F. Jiang a Y.-C. Chung, "Data adapter for querying and transformation between SQL and NoSQL database", *Future Generation Computer Systems*, sv. 65, s. 111–121, pros. 2016, ISSN: 0167739X. DOI: 10.1016/j.future.2016.02.002. WWW: https://linkinghub.elsevier.com/retrieve/pii/S0167739X16300085 (cit. 13. 04. 2019).

[6] D. Mahajan, C. Blakeney a Z. Zong, "Improving the energy efficiency of relational and NoSQL databases via query optimizations", *Sustainable Computing: Informatics and Systems*, sv. 22, s. 120–133, 1. červ. 2019, ISSN: 2210-5379. DOI: 10.1016/j.suscom.2019.01.017. WWW: http://www.sciencedirect.com/science/article/pii/S2210537918301112 (cit. 23. 04. 2019).

[7] L. Gála, J. Pour, Z. Šedivá a Česká společnost pro systémovou integraci, *Podniková informatika: Počítačové aplikace v podnikové a mezipodnikové praxi*. 2015, ISBN: 978-80-247-5457-4.

[8] S. Integrators. (13. dub. 2019). 2018: The IT landscape, Strategic Integrators, WWW: https://strategicintegrators.com/blog/2018-the-it-landscape/ (cit. 14. 04. 2019).

[9] S.-C. Huang, S. McIntosh, S. Sobolevsky a P. C. K. Hung, "Big data analytics and business intelligence in industry", *Information Systems Frontiers*, sv. 19, č. 6, s. 1229–1232, pros. 2017, ISSN: 1572-9419. DOI: 10.1007/s10796-017-9804-9. WWW: https://doi.org/10.1007/s10796-017-9804-9.

[10] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen a S. Belfkih, "Big data technologies: A survey", *Journal of King Saud University - Computer and Information Sciences*, 2017, ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2017.06.001. WWW: http://www.sciencedirect.com/science/article/pii/S1319157817300034.

[11] Adastra. (). Big data nás obklopují, Adastra, WWW: http://www.adastra.cz/clanky/big-data-nas-obklopuji (cit. 14. 01. 2018).

[12] M. Humphries, M. W. Hawkins a M. C. Dy, *Data warehousing: Návrh a implementace*. Praha: Computer Press, 2002, OCLC: 51187218, ISBN: 978-80-7226-560-2.

[13] G. M. Hodge, *Understanding metadata.* Bethesda, Md.: NISO Press, 2004, OCLC: 56196865, ISBN: 978-1-880124-62-8.

[14] (). Metadata, sémantika a sémantický web (vilém sklenák) | ikaros, WWW: https://ikaros.cz/metadata-semantika-a-semanticky-web-vilem-sklenak (cit. 28. 12. 2017).

[15] S. Šimonová, J. Panuš, Univerzita Pardubice a Ekonomicko-správní fakulta, *Databázové systémy i: Pro kombinovanou formu studia*. Pardubice: Univerzita Pardubice, 2007, OCLC: 228501248, ISBN: 978-80-7194-988-6.

[16] S. Williams, *The associative model of data*, 2nd ed. Great Britain: Lazy Software, 2002, 272 s., OCLC: 248255511, ISBN: 978-1-903453-01-8.

[17] C. J. Date, *An introduction to database systems*, 8th ed. Boston: Pearson/Addison Wesley, 2004, 983 s., ISBN: 978-0-321-19784-9.

[18] (13. ún. 2018). Data modeling trends in 2018, DATAVERSITY, WWW: http://www.dataversity.net/data-modeling-trends-2018/ (cit. 09. 03. 2018).

[19] A. Gueidi, H. Gharsellaoui a S. B. Ahmed, "Robotics data real-time management based on nosql solution", in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 2017, s. 131–136. DOI: 10.1109/ICIS.2017.7959982.

[20] D. G. Chandra, "BASE analysis of NoSQL database", *Future Generation Computer Systems*, sv. 52, s. 13 –21, 2015, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2015.05.003. WWW: http://www.sciencedirect.com/science/article/pii/S0167739X15001788.

[21] F. Labs. (). CAP theorem: Its importance in distributed systems, WWW: http://blog.flux7.com/blogs/nosql/cap-theorem-why-does-it-matter (cit. 10. 03. 2018).

[22] (). 6 rules of thumb for MongoDB schema design: Part 1, MongoDB, WWW: https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1 (cit. 17. 03. 2019).

[23] R. Engle, "A METHODOLOGY FOR EVALUATING RELATIONAL AND NOSQL DATABASES FOR SMALL-SCALE STORAGE AND RETRIEVAL", s. 166,

[24] (). What is MongoDB?, MongoDB, WWW: https://www.mongodb.com/what-is-mongodb (cit. 24. 03. 2019).

[25] (). Map-reduce — MongoDB manual, https://github.com/mongodb/docs/blob/v4.0/source/core/map-reduce.txt, WWW: https://docs.mongodb.com/manual/core/map-reduce (cit. 17. 03. 2019).

[26] (). Aggregation pipeline — MongoDB manual, https://github.com/mongodb/docs/blob/v4.0/source/core/aggregation-pipeline.txt, WWW: https://docs.mongodb.com/manual/core/aggregation-pipeline (cit. 24. 03. 2019).

[27] (). Redis, WWW: https://redis.io/ (cit. 24. 03. 2019).

[28] (). Apache cassandra, WWW: http://cassandra.apache.org/ (cit. 24. 03. 2019).

[29] (). Neo4j graph platform – the leader in graph databases, Neo4j Graph Database Platform, WWW: https://neo4j.com/ (cit. 24. 03. 2019).

[30] (). Data warehousing battle of the giants: Comparing the basics of the kimball and inmon models, WWW: http://www.bi-bestpractices.com/view-articles/4768 (cit. 28. 12. 2017).

[31] (). Kimball vs. inmon in data warehouse architecture, WWW: http://www.zentut.com/data-warehouse/kimball-and-inmon-data-warehouse-architectures/ (cit. 28. 12. 2017).

[32] W. H. Inmon, *Building the data warehouse*, 4th ed. Indianapolis, Ind: Wiley, 2005, 543 s., ISBN: 978-0-7645-9944-6.

[33] P. Čech, V. Bureš, Univerzita Hradec Králové a Fakulta informatiky a managementu, *Podniková informatika*. Hradec Králové: Gaudeamus, 2009, OCLC: 428365619, ISBN: 978-80-7041-479-8.

[34] A. J. Oppel, *Databáze bez předchozích znalostí: [průvodce pro samouky*. Brno: Computer Press, 2006, OCLC: 124089576, ISBN: 978-80-251-1199-4.

[35] L. Chen, W. Rahayu a D. Taniar, "Towards near real-time data warehousing", in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, s. 1150–1157. DOI: 10.1109/AINA.2010.54.

[36] G. Bartolini, "Data warehousing with PostgreSQL", s. 48,

[37] (). Historical trend of the popularity ranking of database management systems, WWW: https://db-engines.com/en/ranking_trend (cit. 06. 03. 2018).

Slovník

ANSI American National Standards Institute.

BASE BasicallyAvailable, Soft-state, Eventually Consistent.

HFS Hierarchický souborový systém od firmy IBM.

IoT Internet of Things - internet věcí.

ORDBMS Object relation database management system.

RDBMS Relational database management systems.

SQL Structured Query Language - strukturovaný dotazovací jazyk.

SŘBD Systém řízení báze dat.

Seznam obrázků

Seznam tabulek

Přílohy

- Příloha A - dipl.zip se skládá z těchto souborů:
    1. createDb.sql - skript na vytvoření SQL databáze
    2. nosql.js - skript na nahrání dat do NoSQL databáze
    3. package.json - JavaScript projekt
    4. perf.js - testovací skript
    5. runner.js - spouštěč
    6. sql.js - nahrání dat do SQL databáze

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2017/2018

Studijní program: Systémové inženýrství a informatika
Forma: Kombinovaná
Obor/komb.: Informační management (im2-k)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Bc. Sulíková Kristýna	Borovičky 3164, Dvůr Králové nad Labem	I1500781

TÉMA ČESKY:

Využitelnost NoSQL databází v datových skladech

TÉMA ANGLICKY:

Usability of NoSQL databases in data warehouses solutions

VEDOUCÍ PRÁCE:

Ing. Pavel Čech, Ph.D. - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

SEZNAM DOPORUČENÉ LITERATURY:

http://www.sciencedirect.com/science/article/pii/S1877050917300819
Towards NoSQL-based Data Warehouse Solutions, Procedia Computer Science, Volume 104, 2017, Pages 104-111, Zane Bicevska, Ivo Oditis

http://www.sciencedirect.com/science/article/pii/S2212567115000714
SQL and data analysis. Some implications for data analysits and higher education, Marin Fotache, Catalin Strimbeib a,b Al. I. Cuza University, B-dul Carol 1 nr.22, Iasi, 700505, Romania

http://journals.sagepub.com/doi/full/10.1155/2015/417502
MapReduce Functions to Analyze Sentiment Information from Social Big Data. Ilkyu Ha, Bonghyun Back, Byoungchul AhnFirst Published January 1, 2015

Podpis studenta: ..... **Datum:** .....

Podpis vedoucího práce: ..... **Datum:** .....