

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-68223

**POROVNANIE NOSQL A SQL TYPOV DATABÁZ
BAKALÁRSKA PRÁCA**

2016

Peter Bialeš

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5382-68223

POROVNANIE NOSQL A SQL TYPOV DATABÁZ
BAKALÁRSKA PRÁCA

Študijný program: Aplikovaná informatika
Číslo študijného odboru: 2511
Názov študijného odboru: 9.2.9 Aplikovaná informatika
Školiace pracovisko: Ústav informatiky a matematiky
Vedúci záverečnej práce: Ing. Maroš Čavojský

Bratislava 2016

Peter Bialeš



ZADANIE BAKALÁRSKEJ PRÁCE

Študent: **Peter Bialeš**
ID študenta: 68223
Študijný program: Aplikovaná informatika
Študijný odbor: 9.2.9. aplikovaná informatika
Vedúci práce: Ing. Maroš Čavojský
Miesto vypracovania: Ústav informatiky a matematiky

Názov práce: **Porovnanie NoSQL a SQL typov databáz**

Špecifikácia zadania:

Porovnanie NoSQL a SQL typov databáz a určenie jednotlivých výhod a nevýhod pre konkrétne vybrané druhy databáz.

Úlohy:

1. Naštudovať a zistiť rôzne druhy NoSQL a SQL typov databáz.
2. Oddôvodený výber jednotlivých zástupcov pre jednotlivé typy databáz
3. Vyskúšať prácu s vybranými databázami
4. Zhodnotiť jednotlivé výhody a nevýhody pre vybrané databázy.


Literatúra:

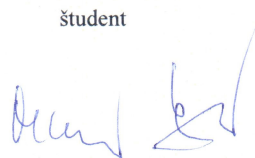
LI, Yishan; MANOHARAN, Sathiamoorthy. A performance comparison of SQL and NoSQL databases. In: Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on. IEEE, 2013. p. 15-19.
CATTELL, Rick. Scalable SQL and NoSQL data stores. ACM SIGMOD Record, 2011, 39.4: 12-27.
BARTHOLOMEW, Daniel. SQL vs. NoSQL. Linux Journal, 2010, 2010.195: 4.

Riešenie zadania práce od: 21. 09. 2015

Dátum odovzdania práce: 20. 05. 2016




Peter Bialeš
študent


prof. RNDr. Otokar Grošek, PhD.
vedúci pracoviska


prof. RNDr. Gabriel Juhás, PhD.
garant študijného programu

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Peter Bialeš
Bakalárska práca:	Porovnanie NoSQL a SQL typov databáz
Vedúci záverečnej práce:	Ing. Maroš Čavojský
Miesto a rok predloženia práce:	Bratislava 2016

Práca sa zaoberá porovnávaním NoSQL a SQL typov databáz, konkrétne MySQL, Elasticsearch a OrientDB. V prvej časti práce sa nachádza analýza problému, pričom sú podrobne uvedené vlastnosti každej databázy, jej výhody a nevýhody a spôsob optimálneho využitia. V druhej časti tejto práce je uvedený návrh riešenia. Návrh riešenia sa zaoberá technológiami, ktoré bolo potrebné použiť pri vykonávaní práce. Táto časť tiež obsahuje popis reprezentácie údajov z MySQL databázy do ostatných typov databáz. V ďalšej časti je popísaná implementácia a podrobný postup konfigurácie jednotlivých databáz v danom prostredí tak, aby vedeli fungovať jedna vedľa druhej. Táto časť taktiež popisuje spôsob nastavenia DreamFactory tak, aby dokázala vytvárať REST API nad MySQL databázou. Obsahuje popis prevedenia importov údajov z MySQL databázy do Elasticsearch a OrientDB. V kapitole overenie riešenia sú prakticky ukázané špecifické príklady funkcionality jednotlivých databáz a ich test, ktorý nad rovnakými údajmi vykonáva rovnaké dopyty v rôznych databázach. V tejto kapitole je znázornená časová odozva jednotlivých databáz pri rôznych selektoch. V závere práce sa nachádza zhrnutie zistených poznatkov, porovnanie s teóriou a vyhodnotenie celej práce.

Kľúčové slová: Databáza, Import údajov, SQL, NoSQL

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Peter Bialeš
Bachelor Thesis:	Comparison of NoSQL and SQL database types
Supervisor:	Ing. Maroš Čavojský
Place and year of submission:	Bratislava 2016

The thesis deals with a comparison of NoSQL and SQL types of databases, specifically MySQL, Elasticsearch and OrientDB. The first part of the thesis is devoted to analysis of issue, where are presented detailed features of each databases, its advantages and disadvantages and optimal ways of using. In the second part solution suggestion is presented. This part focuses on technologies, which were used in implementation of work. It also contains description of data representation from MySQL database to other kinds of databases. In the next part is described implementation and detailed process of configuration of each database in particular environment in a way that the databases can function side by side. This part also describes setting mode of DreamFactory in the way that it can create REST API above the MySQL database. It contains description of transfer of entry imports from MySQL database to Elasticsearch and OrientDB. In the chapter verification of solution are practically shown specific examples of functionality of the databases and their test, which performs equal demands over equal data in various databases. In this chapter is shown time response of the databases at various selects. Conclusion contains summary of detected information, comparison with theory and evaluation of whole thesis.

Keywords: Database, Data import, SQL, NoSQL

Vyhlásenie autora

Podpísaný(á) Peter Bialeš čestne vyhlasujem, že som bakalársku prácu Porovnanie NoSQL a SQL typov databáz vypracoval(a) na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.

Vedúcim mojej bakalárskej práce bol Ing. Maroš Čavojský.

Bratislava, dňa 20.5.2016

.....
podpis autora

Podakovanie

Chcem sa poďakovať vedúcemu záverečnej práce, ktorým bol Ing. Maroš Čavojský, za odborné vedenie, rady a pripomienky, ktoré mi pomohli pri vypracovaní tejto bakalárskej práce.

Obsah

Úvod	10
1 Analýza problému	11
1.1 MySQL Databáza	11
1.2 Elasticsearch	15
1.3 OrientDB	18
2 Návrh riešenia	23
2.1 Použité technológie	23
2.2 Reprezentácia databázy v Elasticsearch	23
2.3 Reprezentácia databázy v OrientDB	24
3 Implementácia	26
3.1 Nastavenie systému a testovacieho prostredia	26
3.2 Nastavenie MySQL a DreamFactory	27
3.3 Nastavenie Elasticsearch a Kibany	28
3.4 Nastavenie OrientDB	29
3.5 Vytváranie Selektov a programovanie webového systému	29
4 Overenie Riešenia	31
4.1 Join selekt pre MySQL	31
4.2 Vyhľadávací selekt pre Elasticsearch	31
4.3 Selekt pre grafovú databázu v OrientDB	33
4.4 Zrovnávací test odozvy jednotlivých databáz	33
4.5 Zhodnotenie	35
Záver	36
Zoznam použitej literatúry	37
Prílohy	I
A Testovacia relačná databáza cohave_v3	II
B Príklad iportovacieho skriptu pre jednu tabuľku z MySQL do Elasticsearch	III

Zoznam obrázkov a tabuliek

Obrázok 1	Náhľad úvodnej obrazovky webového systému	30
Obrázok 2	Reprezentácia grafovej databázy v OrientDB	33
Obrázok 3	Graf porovnanie odozvy MySQL, Elasticsearch, OrientDB	34
Obrázok 4	Graf porovnanie odozvy MySQL a Elasticsearch	35
Tabuľka 1	Tabuľky použité na testovanie	34

Zoznam skratiek a značiek

WWW - World Wide Web

DBMS - Databázový Manažovací Systém

RDBMS - Relačný Databázový Manažovací Systém

Úvod

Databáza je štrukturovaná kolekcia údajov. Od kedy sú počítače veľmi dobré v práci s veľkými dátami, hrajú databázové manažovacie systémy hlavnú úlohu v práci na počítači ako samostatné nástroje alebo ako časti ostatných aplikácií. Dáta uložené v databáze môžu byť hocičo od jednoduchého nákupného zoznamu až po galériu obrázkov alebo obrovské množstvo údajov v korporátnej sieti. Je možné ich rozdeliť do dvoch hlavných kategórií. Na SQL a NoSQL. SQL databázy sú relačné databázy a teda sú to databázy, kde sa používa sada kľúčov na prepojenie jednotlivých tabuliek, ktoré vytvárajú štruktúru SQL databáz. NoSQL databázy nemajú pevne danú štruktúru. V tejto práci sa budeme zaoberať porovnávaním oboch druhov databáz. Budú popísané databázy: MySQL, Elasticsearch a OrientDB. Taktiež bude popísaná práca s týmito databázami, výhody, reprezentácia jednotných údajov na týchto troch typoch databáz.

1 Analýza problému

Spomedzi všetkých databáz boli zvolené tri, ktoré bude táto práca popisovať ďalej. Ako predstaviteľ SQL databáz bola zvolená MySQL databáza. Táto databáza je v súčasnej dobe najpoužívanější, preto bola zvolená. Ako druhá bola zvolená NoSQL databáza Elasticsearch. Je to dokumentová databáza a vyznačuje sa svojou vyhľadávacou schopnosťou. Ako tretia databáza bola zvolená OrientDB. OrientDB je relačná grafová databáza, no ponúka aj výhody dokumentovej databázy.

1.1 MySQL Databáza

MySQL je v súčasnej dobe najpopulárnejší Open Source SQL databázový manažovací systém. Je vyvíjaný a distribuovaný korporáciou Oracle. SQL, konkrétne MySQL je štruktúrovaný dopytovací jazyk. SQL je najbežnejší štandardizovaný jazyk používaný na prístup do databáz. V závislosti od programovacieho prostredia, sa dá do SQL vstúpiť priamo (napr. generovanie reportov), vkladať SQL reťazce do zdrojového kódu napísaného v inom jazyku, alebo použiť language-specific API, ktorá skrýva SQL syntax. SQL je definované ANSI/ISO SQL štandardom. SQL štandard sa vyvíjal od roku 1986 a existuje už niekoľko verzií [5].

Výhody / Nevýhody MySQL

Prenosnosť

- Napísaný v C a C++
- Testovaný s veľkým okruhom rôznych kompilérov
- Pracuje na rôznych platformách
- Pre prenosnosť používa CMake. Staršie verzie používajú GNU Automake, Autoconf a Libtool
- Používa viacvrstvový návrh serveru s nezávislými modulmi
- Bol dizajnovaný viacvláknovo na používanie vo viacvláknových jadrách, na ľahké použitie s viacjadrovými procesormi.
- Poskytuje transakčné a netransakčné ukladacie enginy.

- Používa veľmi rýchle B-tree diskové tabuľky (MyISAM) s kompresiou indexov.
- Bol vyrobený na relatívne rýchle pridávanie ďalších úložiskových enginov. (Toto je dôležité pre poskytovanie SQL rozhrania pre domáce databázy.)
- Používa veľmi rýchly vláknovo založený pamäťový alokovací systém.
- Vykonáva veľmi rýchlo joiny použitím optimalizovaného spájania vnorených joinov
- Implementuje vnútro-pamäťové hashovacie tabuľky, ktoré sú využité ako dočasné tabuľky
- Implementuje SQL funkcie použitím vysoko optimalizovaných knižníc tried, ktoré môžu byť rýchle ako sa len dá. Často sa stáva, že v tomto prípade nieje potrebná alokácia pamäte po inicializácii dotazu.
- Poskytuje server ako oddelený program pre použitie v klient/server sieťovom prostredí a knižnicu, ktorá môže byť nalinkovaná do samostatných aplikácií.

Bezpečnosť

- Privilegovaný a heslový systém je veľmi flexibilný a bezpečný, a dovoľuje zapnúť host-based verification
- Zabezpečenie heslom je šifrované od kedy sa užívateľ pripojí na server

Škálovateľnosť a obmedzenia

- Podpora pre veľké databázy. MySQL Server sa používa s databázami, ktoré obsahujú 50 miliónov záznamov. Taktiež existujú užívatelia, ktorí používajú MySQL Server na hostovanie 200000 tabuliek a 5 000 000 000 riadkov
- Má podporu pre 64 indexov na jednu tabuľku. Každý index môže pozostávať od 1 až po 16 stĺpcov. Maximálna šírka indexu je 767 bajtov pre InnoDB tabuľky, alebo 1000 bajtov pre MyISAM.

Konektivita

- Klient sa môže pripojiť na MySQL Server pomocou viacerých protokolov:
 - Použitím TCP/IP socketov na akejkoľvek platforme

- Na systémoch Windows sa môžu klienti pripojiť taktiež aj použitím named pipes ak je server štartovaný použitím prepínača *-enable-named-pipe*. Windows servery poskytujú taktiež pripojenie prostredníctvom zdieľanej pamäte ak sa server štartuje s prepínačom *-shared-memory*. Klienti sa môžu pripojiť cez zdieľanú pamäť použitím prepínača *-protocol=memory*.
 - Klienti na Unixových systémoch sa môžu pripojiť použitím Unix domain socket súborov.
- MySQL klientske programy môžu byť napísané vo viacerých jazykoch. Knižnica klienta, ktorá je naprogramovaná v C je dostupná pre klientov programovaných v C a C++, alebo pre jazyk, ktorý poskytuje C bindings.
 - API je dostupné pre C, C++, Eiffel, Javu, Perl, PHP, Python, Ruby a TCL. Connector/ODBC (MyODBC) rozhranie poskytuje podporu pre viacero programov, ktoré používajú ODBC pripojenia. Connector/ODBC má dostupné zdrojové kódy.
 - Connector/J rozhranie poskytuje MySQL podporu pre Javovské klientské programy, ktoré používajú JDBC pripojenia. Klienti sa dajú spustiť na operačnom systéme Windows alebo Unixe. Zdrojové kódy pre Connector/J sú takisto dostupné.
 - MySQL Connector/Net umožňuje vývojárom jednoduchým spôsobom vytvárať .NET aplikácie, ktoré potrebujú bezpečnú, vysoko výkonnú dátovú konektivitu s MySQL. Takisto implementuje potrebné ADO.NET rozhrania a integruje sa do ADO.NET aware tools. Vývojári môžu vytvárať aplikácie použitím .NET jazykov. MySQL Connector/Net je plne manažovateľný ADO.NET driverom naprogramovaným v C#.

Lokalizácia

- Server vie poskytovať errorové hlášky klientom vo viacerých jazykoch
- Plná podpora pre rôzne znakové sady, napríklad: latin1 (cp1252), german, big5, ujis...
- Všetky dáta sú uložené vo vybranej znakovej sade
- Triedenie a porovnávanie sa vykonáva podľa zvolenej znakovej sady a radenia. Toto je možné zmeniť keď MySQL server štartuje.

- Casová zóna servera sa môže dynamicky meniť a individuálni klienti si môžu špecifikovať ich vlastnú časovú zónu.

Klienti a Nástroje

- MySQL zahŕňa niekoľko klientských programov a utilít. Tieto programy zahŕňajú obe konzolové programy ako sú mysqldump a mysqladmin a grafické programy ako MySQL Workbench.
- MySQL Server má vstavanú podporu pre SQL príkazy slúžiace na kontrolu, optimalizáciu a opravu tabuliek. Tieto príkazy sú dostupné z príkazového riadku cez mysqlcheck klienta. MySQL taktiež prikladá myisamchk, veľmi rýchly konzolový nástroj na vykonávanie týchto operácií na MyISAM tabuľkách.
- MySQL programy môžu byť volané s parametrom *-help* alebo *-?* Pre získanie online podpory.[6]

Použitelnosť

MySQL server môže plynulo bežať na desktope aj notebooku popri aplikáciach, webových serveroch, atď. Vyžaduje si malú, alebo takmer žiadnu pozornosť. Ak administrátor vyhradí celý počítač na MySQL, môže zmeniť nastavenia aby využil celú pamäť, procesor a vstupno-výstupné interfejsy. MySQL sa dá škálovať na clustery, ktoré sú vzájomne prepojené cez sieť. MySQL server bol originálne vyvinutý na rýchlejšiu prácu s veľkými databázami oproti existujúcim riešeniam a bol úspešne využívaný vo vysoko náročných produkčných prostrediach po dlhé roky. Aj keď je stále vo vývoji, MySQL Server ponúka dnes už bohaté a užitočné funkcionality. Medzi ne patrí konektivita, rýchlosť, ktoré robia MySQL Server veľmi situovaný pre prístup k databázam na internete.

MySQL databázový softvér je klient/serverový systém pozostávajúci z viac vlákňového SQL Servera, ktorý podporuje rôzne backendy, niekoľko rôznych klientov a knižníc, administratívnych nástrojov, a širokého okruhu rozhraní pre programovanie aplikácií (API). MySQL Server sa taktiež poskytuje aj ako viac vlákňová knižnica, ktorá sa dá linkovať do aplikácií pre dosiahnutie menších, rýchlych a ľahko manažovateľných samostatných produktov.[5]

1.2 Elasticsearch

Elasticsearch je open-source vyhľadávací engine postavený na technológii Apache Lucene™ full textového vyhľadávacieho enginu. Lucene je pravdepodobne najpokročilejší, vysoko výkonný a plne vybavený vyhľadávacou knižnicou, ktorá je tiež open-source. Elasticsearch je naprogramovaný v Jave a používa Lucene interne pre všetku jeho indexáciu a vyhľadávanie, ale kladie si za cieľ vytvoriť full-textový vyhľadávač so zjednodušeným, súdržným RESTful API, kde sú skryté zložité funkcie Lucene.[3]

Výhody / Nevýhody Elasticsearch

Elasticsearch je oveľa viac ako full-textový vyhľadávač. Dal by sa popísať nasledovnými spôsobmi:

- distribuované real-time úložisko dokumentov, každé pole je indexované a je možné ho vyhľadať
- distribuovaný vyhľadávací engine s real-time analýzami
- schopný škálovania stoviek serverov a petabajtov štrukturovaných a neštrukturovaných dát

Balíky so svojou funkcionalitou sa nachádzajú na osobitnom serveri, takže klientská aplikácia môže komunikovať pomocou RESTful API rozhrania.

Je veľmi jednoduché začať prácu s Elasticsearch. Elasticsearch je od výroby dodávaný s prednastavenými konfiguračnými súbormi tak, aby skryl komplikovanú teóriu pred začiatočníkmi. Pracuje hneď po nainštalovaní. Užívateľ vie byť produktívny aj s minimálnymi vedomosťami.

Ako rastú vedomosti užívateľa o Elasticsearch, tým viac môže konfigurovať zložitejšie nastavenia. Celý engine Elasticsearch je konfigurovateľný a flexibilný.

Elasticsearch sa dá používať a modifikovať zadarmo. Je dostupný pod Apache 2 licenciou. Zdrojové kódy sú dostupné na GitHubu.[3]

Elasticsearch je takmer real-timeová platforma. Má nízku latenciu (okolo sekundy) od času kedy sa dokument zaindexuje a časom, kedy sa stane vyhľadateľným v databáze. Pozostáva z nasledujúcich častí:

Cluster

Cluster je kolekcia jedného, poprípade viacerých uzlov. Cluster je definovaný unikátnym

výrazom, ktorý je defaultne nastavený na “elasticsearch”. Tento výraz je dôležitý, lebo uzol môže byť súčasťou clusteru ak je uzol nastavený na spojenie s clusterom cez výraz. V rôznych prostrediach je dôležité nepoužívať znovu už použité názvy clusterov, inak by sa mohlo stať, že uzly by sa mohli spojiť s nesprávnym clusterom. Vo väčšine prípadov sa doporučuje mať cluster len s jedným uzlom. Okrem toho sa dá mať viac nezávislých clusterov s jedinečnými názvami clusterov.

Uzol

Uzol je jednoduchý server, ktorý je súčasťou clusteru, uchováva v sebe dáta a spolupodieľa sa na clusterových indexovaniach a vyhľadávaciach schopnostiach. Ako cluster, tak aj uzol je identifikovaný názvom, ktorý je od výroby nastavený na náhodné meno Marvel postavy, ktorá je pridelená uzlu pri štarte služby. Dá sa takisto definovať aj vlastný názov uzlu ak užívateľovi nevyhovuje prednastavený. Tento názov uzlu je dôležitý pre administratívne účely, kde užívateľ chce identifikovať, ktoré servery v sieti korešpondujú s ktorými uzlami v Elasticsearch clusteri.

Uzol môže byť konfigurovaný na prepojenie so špecifickým clusterom. Od výroby je každý uzol nastavený aby sa spájal s clustrom pomenovaným “elasticsearch”, čo znamená, že ak si užívateľ založí niekoľko uzlov s predpokladom, že sa môžu vidieť navzájom, tieto uzly budú automaticky spojené do jedného clusteru nazvaného “elasticsearch”.

V samostatnom clustri sa môže nachádzať neobmedzený počet uzlov. Okrem toho, ak sú spustené na sieti ďalšie Elasticsearch uzly, spustenie samostatného uzlu bude podľa predvoleného postupu nový samostatný cluster s názvom elasticsearch.

Index

Index je kolekcia dokumentov, ktoré majú trochu podobné charakteristiky. Napríklad máme index na údaje zákazníkov, index na katalóg produktov a ďalší index na dáta. Index je definovaný názvom (musí mať malé písmená) a jeho názov je používaný na odkazovanie sa na index, keď prebieha indexovanie, prehľadávanie, update a mazanie dokumentov, ktoré index obsahuje.

V samostatnom indexe sa dá ďalej definovať viacero indexov.

Typ

Vo vnútri indexu sa dá definovať jeden, alebo viac typov. Typ je logická kategória/partícia indexu, ktorého schéma závisí len na užívateľovi, ktorý ju vytvára. Vo všeobecnosti, typ je definovaný pre dokumenty, ktoré majú niektoré polia spoločné. Napríklad pri blo-

govacej platforme, ktorej dáta sú ukladané na osobitnom indexe. V tomto indexe môžu byť definované typy pre užívateľské dáta, ďalšie typy pre dáta blogovacieho systému a ešte ďalší typ pre komentáre.

Dokument

Dokument je základný prvok informácie, ktorú je možné indexovať. Pre lepšiu predstavu, majme dokument pre jedného užívateľa, dokument pre jeden produkt a dokument jedného predávateľa. Tieto dokumenty sú reprezentované vo formáte JSON, ktorý je vo veľkej miere rozšírený formát pre výmenu dát.

Vo vnútri indexu alebo typu sa dá uchovávať toľko dokumentov, koľko užívateľ potrebuje uchovať. Dokument je fyzicky umiestnený v indexe a musí byť indexovaný alebo priradený typu vo vnútri indexu.

Shardy a Repliky

Index dokáže teoreticky uchovať veľké množstvo údajov ktoré môžu prekročiť hardvérové limity jedného uzlu. Pre lepšie pochopenie, napríklad jeden index pozostávajúci z miliárdov dokumentov, ktorý zaberá 1TB na disku sa nemusí vojsť na disk jedného uzla, alebo uzol sa môže stať pomalým na odozvu, keď sa naň začne dopytovať veľa užívateľov.

Na vyriešenie tohto problému ponúka Elasticsearch funkcionality, ktorá umožňuje ďalej deliť index na viacero častí nazývaných shardy. Pri vytváraní indexu sa dá zadať, koľko shard-ov chce mať užívateľ k dispozícii. Každý shard je sám o sebe plne funkčný a nezávislý “index”, takže môže byť hostovaný na hociktorom uzle v clusteri.[1]

Použitelnosť

Tu je zopár príkladov použitia Elasticsearch:

- Využitie ako webový obchod kde si môžu užívatelia vyhľadávať produkty, ktoré sú určené na predaj. V tomto prípade sa dá Elasticsearch využiť ako úložisko celých katalógov produktov a inventáru a poskytovať vyhľadávanie a automatické dopĺňanie vyhľadávaných produktov.
- Dá sa využiť ako zberač logov alebo transakčných dát a následne môže analyzovať a minovať tieto údaje na reprezentáciu trendov, štatistík, sumarizácií alebo anomálií. V tomto prípade sa dá využiť Logstash (súčasť Elasticsearch/Logstash/Kibana balíčka) na zber, zoskupovanie, parsovanie dát, a potom pomocou Logstash-u naplniť týmito údajmi Elasticsearch. Keď sú už raz dáta v Elasticsearch, dá

sa používať vyhľadávanie a zoskupovanie na získavanie informácií ktoré zaujímajú zákazníka.

- Využitie ako upozorňovač na ceny, ktorý umožňuje zákazníkom, ktorí zvyknú hľadiť na ceny produktov aby si špecifikovali pravidlá ako napr: “Zaujímam sa o kúpu špecifickej elektroniky a chcem byť upozornený keď sa cena za mesiac konkrétnej elektroniky pohybuje nižšie ako mnou zadaná hodnota”. V tomto prípade sa dajú znižovať výrobcove ceny tým, že ich nahráme do Elasticsearch a využijeme reverznej hľadacej schopnosti na porovnávanie pohybu cien voči požiadavkam zákazníkov a prípadne pošleme upozornenia zákazníkovi, keď sa našli zhody.
- Dá sa využiť na analytické/business-intelligence potreby a potreby rýchleho investovania, analýzy, vizualizácie a vykonávanie ad-hoc dotazov na veľké množstvo údajov (myslené milióny a miliardy údajov). V tomto prípade sa dá Elasticsearch využiť na uloženie dát a potom pomocou Kibany vytvoriť vlastné dashboard-y, ktoré môžu vizualizovať vaše dáta, ktoré sú pre vás dôležité. Dalej sa dá využiť zoskupovacia funkcionality Elasticsearch na vytvorenie kompletných dotazov analytických nástrojov na vaše dáta.[2]

1.3 OrientDB

OrientDB je druhá generácia distribuovanej grafovej databázy s flexibilitou dokumentov v jednom produkte. Je dodávaná ako open-source s komerčnou Apache 2 licenciou. OrientDB bola vyvíjaná od základu s kladením dôrazu na výkonnosť. Je veľmi rýchla pri zapisovaní aj čítaní a vie uchovať 400 tisíc záznamov za sekundu na bežnom hardvéri. Pri databáze zloženej z dokumentov reprezentuje vzťahy medzi nimi ako priame prepojenia v grafovej databáze. Je možné prechádzať časťami jednotlivých vetiev a grafami behom pár milisekúnd. OrientDB podporuje bezschémový, schémový, alebo hybridný mód pre dátový model. Taktiež má silný bezpečnostný systém profilovania založený na užívateľoch a roliach a taktiež podporuje “horizontálnu” bezpečnosť na každý záznam v databáze osobitne. OrientDB bez problémov podporuje OAuth tikety, Kerberos a LDAP autentifikáciu. Taktiež podporuje rozšírený SQL dialekt, vďaka čomu je veľmi rýchla a ľahká na použitie pre tých, ktorí sú skúsení v relačných databázach. OrientDB podporuje multi-Master a Sharded architektúru, takže všetky servery sú master servery. Táto funkcionality poskytuje horizontálnu škálovateľnosť a spoľahlivosť.[4]

Výhody / Nevýhody OrientDB

- vytvorená pre rýchlosť

OrientDB bola vytvorená od základov s dôrazom na výkonnosť. Je rýchla aj počas zápisu aj počas načítavacích operácií. OrientDB dokáže vkladať dokumenty ako každá iná dokumentová databáza, ale taktiež podporuje aj vzťahy medzi nimi. Nepoužíva na systémové prostriedky náročný JOIN, ale používa veľmi rýchle perzistentné pointre medzi záznamami, ktoré sa využívajú vo svete grafových databáz. Je možné prechádzať časti, alebo celé stromy a grafy záznamov behom niekoľkých milisekúnd. Dokumenty je možné rozdeliť, pričom ostanú prepojené pomocou ID záznamu, tzv. “Record ID”. Vďaka tomuto sa môžu dokumenty ľahko prepájať, čo je výhodnejšie ako by sa mali pokaždé vkladať celé. Výsledná databáza je rýchlejšia, menšia a prehľadnejšia s lepším využitím RAM a efektívnejším ukladaním do cache pamäte. Po načítaní stromu dokumentov OrientDB zostavuje kompletnú štruktúru dokumentu a načítava všetky prepojenia transparentne. Čím viac dokumentov pribúda v relačných DBMS, tým viac sú tieto systémy pomalé. Je to aj vďaka Joinom, ktoré majú veľké systémové požiadavky na svoj beh. Namiesto toho OrientDB spracováva vzťahy ako fyzické odkazy na záznamy, ktoré sú pridané len raz, a to pri vytváraní hrany “edge” (zložitosť $O(1)$). Pre porovnanie s RDBMS, ktorý prepočítava vzťahy pri každom dopyte do databázy má zložitosť $O(\log N)$. S OrientDB je rýchlosť ako sa vykoná dotaz neovplyvnená veľkosťou databázy. Je vždy konštantná. Toto je dôležité pri prevádzke Big Data.

- Bezkonkurenčná flexibilita

Výmena DBMS môže stať veľa času a prostriedkov, no OrientDB výrazne znižuje potrebu mať viac produktov na dosiahnutie rovnakého cieľa. Napríklad, OrientDB je grafová a dokumentová databáza zároveň. Vyznačuje sa tým, že podporuje schémový, bezschémový a hybridný dátový model. Zabezpečenie je na úrovni jednotlivých záznamov. Taktiež podporuje SQL syntax, takže ju dokážu prakticky hneď používať aj ľudia, ktorí ovládajú SQL. Má podporu Multi-Master replikácie a taktiež shardingu. OrientDB sa vyznačuje veľkou škálovateľnosťou s automatickou konfiguráciou. Taktiež podporuje natívne HTTP REST protokol s JSON štruktúrou.

- Nulová konfigurácia, Multi-Master architektúra

OrientDB podporuje Multi-Master + Sharded Architektúru, čo v praxi znamená, že všetky servery sú v mode master. Táto vlastnosť poskytuje horizontálnu škálovateľnosť

a spoľahlivosť. elastická lineárna škálovateľnosť. Pri master-slave architektúre sa master často stáva prekážkou. S OrientDB nieje priepustnosť obmedzená na jediný server. Globálna priepustnosť je v tomto prípade súčet priepustností všetkých serverov. Pokiaľ treba navýšiť počet uzlov v sieti, stačí jednoducho pridať uzol do siete a on sa automaticky pripojí ku existujúcemu distribuovanému serverovému clusteru bez nejakej ďalšej konfigurácie. Synchronizácia prebieha automaticky, akonáhle je server online. Distribuovaná spoľahlivosť Väčšina riešení NoSQL sú používané ako “cache” pamäť na zrýchlenie nektorých prípadov použitia, zatiaľ čo hlavné databázy zostávajú v relačnej databáze DBMS. Pre tento dôvod je priemerný NoSQL systém postavený viac na výkone a škálovateľnosti ako na spoľahlivosti. OrientDB však nieje priemerný NoSQL DBMS. Ak sa serverový uzol zrúti, tak OrientDB je schopný obnoviť obsah databázy po havárii. Všetky nevybavené transakcie sú automaticky vrátené späť. Serverový cluster prerozdeľuje záťaž okamžite naprieč dostupnými uzlami a všetkých klientov, ktorí sú pripojení na uzol, ktorý zlyhá automaticky prepojí na dostupný serverový uzol s nulovou chybovosťou na aplikačnej úrovni. Perfektné cloud dátázové riešenie Vzhľadom k uvedeným výhodám je OrientDB ideálny pre cloud. Stovky serverov môžu zdieľať záťaž, je možná vodorovná škálovateľnosť naprieč modernými distribuovanými dátovými centrami.

- Jednoduchá inštalácia a používanie

OrientDB server je celý napísaný v Jave a je možné ho spustiť na ľubovoľnej platforme bez konfigurácie a inštalácie. Plná verzia distribúcie servera má okolo 2 MB bez databázy. Pokiaľ ide o otázku jazykov, tak OrientDB sa zameriava na štandardy. Používa SQL syntax, ktorá je doplnená o grafovú funkcionálnosť. Prípadne dá sa použiť rozhranie API Thinkerpop, ktoré je štandardom u väčšiny grafových databáz. Pracovať s OrientDB sa dá naučiť v priebehu pár hodín. Na stránke OrientDB je podrobný manuál, taktiež je tam aj viac ako 2000 členná komunita užívateľov po celom svete, ktorí využívajú OrientDB a vedia pomôcť pri riešení prípadných problémov.

- Nízke cenové náklady na licencie

Pri používaní OrientDB Community Edition niesú žiadne náklady. OrientDB je distribuovaná pod licenciou Apache 2 Open Source, ktorá je najviac liberálna spomedzi všetkých a dovoľuje používať OrientDB zadarmo na akýkoľvek účel. Vďaka bohatej škále funkcií, multimodelovému dizajnu, OrientDB znižuje potrebu vlastniť viac produktov nadosiahnutie rovnakých cieľov. Jednoduchosť použitia, nulová konfigurácia multi-master architektúry a SQL syntax robia OrientDB ideálnou voľbou na výber.

- Ideálna pre podnik

Zatiaľ čo väčšina NoSQL databáz sa používajú ako sekundárne databázy, OrientDB je silný a dostatočne flexibilný na využitie ako prevádzkového databázového systému. Nevyžaduje si reštart, alebo vypnutie na údržbu, vie automaticky použiť voľné miesto získané z odstránených záznamov a pritom vie byť stále online. OrientDB Enterprise Edition je komerčný softvér vytvorený ako vrchol Community Edition, bol vyvíjaný rovnakým tímom, ktorý vyvíjal engine OrientDB. Táto edícia slúži na rozšírenie prémiových funkcionalít, ktoré sa nenachádzajú už v Community Edition. Enterprise Edition je zadarmo aj s podporou.

- Jedinečná na trhu

Niž dostupné na trhu sa neblíži ku OrientDB, čo sa týka ponuky funkcionality a ceny. Na dosiahnutie rovnakých výsledkov s OrientDB by bolo treba niekoľko rôznych konkurenčných DBMS produktov, no použitím OrientDB neterba mať viac produktov.

- Je Open Source

Zdrojový kód OrientDB je verejný a transparentný, čiže každý môže kontrolovať, testovať, reportovať a riešiť problémy. Oproti databázam, ktoré boli vyvíjané hárstvou developerov a testované pár testermi má OrientDB veľku prevahu.[9]

Použitelnosť

Prípady, kde sa dá použiť OrientDB je mnoho, no na týchto dvoch príkladoch vidno jednoduchosť a silu OrientDB.

- dlhodobé pozorovania

OrientDb sa dá využiť na správu záznamov týkajúcich sa historických informácií. Ak je dostupných milión záznamov, vtedy sa počet indexov začína pohybovať na hranici, lebo u bežnej relačnej databázy je zložitosť algoritmu pre nájdenie záznamu $O(\log N)$. Toto je hlavný dôvod prečo sa pri veľkom počte záznamov používa OrientDB. Ak sú k dispozícii milióny záznamov, tak najlepším riešením je lineárne škálovanie, kedy sa dá vyhnúť využívaniu indexov a to vďaka využitiu vlastností grafovej databázy.[8]

- chat

OrientDB je ideálny na použitie v chatovacej aplikácii, kde sa dá pomocou document API vytvoriť jedna trieda pre chatovaciu miestnosť bez použitia indexu a teda týmto postupom sa dá získať rýchly prístup k posledným správam, nakoľko nové záznamy majú automaticky generovaný @rid tak aby sa jeho hodnota navyšovala a teda podľa triedy a @rid sa dá zistiť obsah posledných správ.[7]

2 Návrh riešenia

Ako testovacie dáta boli zvolené dáta z databázy cohove v3 s najnovšími údajmi zo Septembra 2015, ktorá slúžila na zber údajov z mobilných zariadení určených na mobile computing. Dané dáta boli reprezentované v relačnej databáze MySQL a bolo potrebné ich importovať aj do ostatných druhov databáz.

Daná databáza obsahovala záznamy o mobilných zariadeniach, o ich polohe, ďalej údaje o dostupných wifi sieťach v lokalite kde sa užívateľ nachádzal a údaje o používateľovi.

V prílohe A je uvedený zjednodušený model databázy z ktorej sa vychádzalo. Pôvodná databáza obsahovala aj tabuľky, ktoré neboli viazané na ostatné dáta žiadnym kľúčom. Tieto tabuľky boli z diagramu pre prehľadnosť vynechané.

2.1 Použité technológie

Na otestovanie skorej uvedených typov databáz si bolo potrebné zadovážiť inštalačné súbory jednotlivých databáz a nainštalovať ich na predom pripravené prostredie. Prostredie, v ktorom sa inštalovali a testovali databázy je virtuálny stroj, na ktorom je nainštalovaný systém Linux, konkrétne distribúcia Ubuntu 14.04 LTS. K tomu aby sa dalo plnohodnotne testovať vybrané typy databáz bolo potrebné mať k dispozícii nainštalovaný web server, php kompilátor a sql databázu. Ideálnym riešením bolo použiť LAMP Stack, ktorý zahŕňa Apache server, PHP5, MySQL s phpMyAdmin. Takisto bol nainštalovaný DreamFactory Stack, ktorý má za úlohu vytvárať RESTfull rozhranie pre rôzne typy databáz, no v tomto prípade bol použitý pre MySQL databázu, aby mohli byť všetky typy databáz porovnateľné cez RESTfull rozhranie rovnako porovnateľné.

2.2 Reprezentácia databázy v Elasticsearch

Daná databáza je reprezentovaná v Elasticsearch ako sada dokumentov, ktoré sa nachádzajú na jednom uzle. Tieto dokumenty sú indexované na základe jednotlivých riadkov v tabuľkách. Za účelom rýchlejšieho indexovania bol vytvorený konfiguračný súbor pre uzol, ktorý obsahuje dátové typy jednotlivých položiek dokumentov. Všetky dátové typy boli zachované, okrem položiek “latitude” a “longitude” v dokumente “funfi_location”, ktoré boli nahradené dátovým typom “geo_point”, nakoľko ide o geografické súradnice. Po nastavení mapovania však importovací skript, ktorý bol napísaný v php neimportoval do Elasticsearch žiaden údaj. Bolo to spôsobené tým, že importovací skript importoval údaje z MySQL databázy do Elasticsearch vo formáte String. Údaje

boli napokon do databázy nainportované vo formáte Stringov. Vzor pre import tabuľky “funfi_wifi_location” ako dokumentu do Elasticsearch vidno na ukážke nasledujúceho zdrojového kódu:

PUT data

```
{
  "mappings": {
    "funfi_wifi_location": {
      "properties": {
        "_id": {"type": "integer"},
        "wifiNetwork_id": {"type": "integer"},
        "location_id": {"type": "integer"},
        "RSSI": {"type": "integer"},
        "signalLevel": {"type": "integer"},
        "sensor_time": {"type": "date", "format": "epoch_second"},
        "log_time": {"type": "date", "format": "epoch_second"},
        "inserted_time": {"type": "date",
                          "format": "yyyy-MM-dd HH:mm:ss"},
        "device_id": {"type": "integer"},
        "app_version": {"type": "string"}
      }
    }
  }
}
```

2.3 Reprezentácia databázy v OrientDB

Reprezentácia danej databázy v OrientDB bola realizovaná ako grafová databáza, pričom každý jej vrchol tvoria jednotlivé dokumenty. Dátové typy jednotlivých položiek dokumentov boli nainportované z MySQL automaticky pomocou importovacieho skriptu a ostali nezmenené. Aby sa odlíšilo z akej tabuľky boli dokumenty importované, tak im boli priradené triedy podľa názvov tabuliek v MySQL.

Príklad importovacieho skriptu možno vidieť na nasledujúcich riadkoch:

```
{
  "config": {
    "log": "debug"
  },
  "extractor": {
    "jdbc": {
      "driver": "com.mysql.jdbc.Driver",
      "url": "jdbc:mysql://localhost/data",
      "userName": "root",
      "userPassword": "root",
      "query": "select * from general_device_log"
    }
  },
  "transformers" : [
    { "vertex": { "class": "general_device_log"} }
  ],
  "loader": {
    "orientdb": {
      "dbURL": "plocal:../databases/data",
      "dbUser": "admin",
      "dbPassword": "admin",
      "dbAutoCreate": true,
      "tx": false,
      "batchCommit": 1000,
      "dbType": "graph"
    }
  }
}
```

3 Implementácia

3.1 Nastavenie systému a testovacieho prostredia

Ako bolo už spomínané, všetky testované typy databáz boli nainštalované na virtuálnom stroji. Potreba mať virtuálny stroj vznikla na základe potreby vedieť koľko pamäte a prostriedkov potrebujú jednotlivé druhy databáz. Ako virtualizačný nástroj bol použitý VM VirtualBox verzia 4.3.12 od firmy Oracle. Pri výbere operačného systému bolo potrebné brať ohľad na spoľahlivosť, rýchlosť, manažovateľnosť, a ďalšie faktory. Tieto podmienky spĺňa zrejme najlepšie Linux, preto ako operačný systém bol použitý tento systém. Distribúcia Linuxu bola zvolená Ubuntu 14.04 LTS kôli veľkej užívateľskej podpore.

Po nainštalovaní systému sa inštaloval LAMP Stack, ktorý v sebe zahŕňa tieto servery a funkcionality: Apache, MySQL, PHP, phpMyAdmin. Inštalácia bola uskutočnená pomocou nasledujúcich príkazov:

```
sudo apt-get update
sudo apt-get install apache2
sudo apt-get install mysql-server php5-mysql
sudo mysql_install_db
sudo mysql_secure_installation
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
sudo vi /etc/apache2/mods-enabled/dir.conf
```

Obsah konfiguračného súboru dir.conf bol prepísaný nasledovným kódom:

```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.htm
</IfModule>
```

Po prepísaní dir.conf nasledovali tieto príkazy:

```
sudo service apache2 restart
sudo apt-get install phpmyadmin
sudo service apache2 restart
```

3.2 Nastavenie MySQL a DreamFactory

Importovaná databáza mala veľkosť 1,8 GB a importovanie v phpMyAdmin nebolo možné kôli jej veľkosti. Preto bolo nutné zmeniť v súbore php.ini nasledujúce parametre:

```
upload_max_filesize=2000M
post_max_size=2000M
max_execution_time=1800
session.gc_maxlifetime=3600
```

Vďaka týmto parametrom bolo možné importovať databázu do MySQL. Po importovaní databázy do MySQL bolo potrebné vytvoriť nad ňou REST aplikačné rozhranie, aby bolo možné sa na ňu dotazovať rovnakým spôsobom ako na ostatné druhy databáz. Na tento účel bol nainštalovaný DreamFactory Stack, ktorý má grafický inštalátor a nebolo potrebné ho inštalovať cez príkazový riadok. DreamFactory Stack obsahuje v sebe aj HTTP server s MySQL databázou. Po inštalácii bolo potrebné prejsť DreamFactory s MySQL databázou, ktorá obsahovala naimportované údaje. To však nebolo možné spraviť podľa návodu, ktorý sa uvádza na oficiálnom webe Dream Factory. Preto sa využila vstavaná MySQL databáza priamo v DreamFactory. Na import údajov do nej bolo potrebné zmeniť tieto konfiguračné súbory:

dreamfactory-2.1.0-4/apps/phpmyadmin/conf/httpd-app.conf

Vnútri tohto konfiguračného súboru bolo potrebné zmeniť tieto parametre:

```
<IfModule php5_module>
    php_value upload_max_filesize 2000M
    php_value post_max_size 2000M
</IfModule>
```

dreamfactory-2.1.0-4/php/etc/php.ini

V tomto súbore bolo treba zmeniť tieto riadky:

```
max_execution_time=1800
session.gc_maxlifetime=3600
```

A v súbore dreamfactory-2.1.0-4/apps/phpmyadmin/htdocs/config.inc.php bolo potrebné zmeniť tento riadok:

```
$cfg[ 'ExecTimeLimit' ] = 1800;
```

Po absolvovaní týchto krokov, bola MySQL Databáza pripravená na import údajov. Po importe údajov cez phpMyAdmin (<http://localhost:8080/phpmyadmin>) bolo vidno vo

webovom rozhraní DreamFactory databázu aj s údajmi. Na záložke API Docs sa dali získať url odkazy na prácu s databázou.

Nasledovným krokom bolo nainštalovať prostredie Java Runtime Environment, ktorý je potrebný aby sa mohol spustiť databázový server pre Elasticsearch a OrientDB. Inštalácia bola vykonaná pomocou nasledujúceho príkazu:

```
sudo apt-get install default-jre
```

3.3 Nastavenie Elasticsearch a Kibany

Po inštalácii Java Runtime Environment sa rozbalili inštalačné balíky pre Elasticsearch (v. 2.0.0), Kibana (v. 4.2.0-linux-x64) a OrientDB (edícia community v. 2.1.9) do domovského adresára.

Nasledovala inštalácia pluginu Sense pre Kibanu v ktorom sa dajú editovať a vytvárať skripty pre Elasticsearch. Inštalácia Sense bola vykonaná pomocou nasledujúcich príkazov:

```
cd ../kibana-4.2.0-linux-x64/  
bin/kibana plugin --install elastic/sense
```

Potom bolo treba nakonfigurovať Elasticsearch nasledovným spôsobom. Bolo treba otvoriť súbor `elasticsearch-2.3.1/config/elasticsearch.yml` a pridať doň nasledujúce riadky:

```
cluster.name: mcomputing  
node.name: dates  
bootstrap.mlockall: true  
network.host: 127.0.0.1
```

Nasledovným krokom bolo spustenie serveru Elasticsearch a import údajov z MySQL. Pre tieto potreby boli vyvinuté skripty v PHP, pomocou ktorých bolo možné previesť import z MySQL do Elasticsearch. V prílohe B je uvedený príklad importovacieho skriptu pre jednu tabuľku. Pred tým však bolo potrebné vytvoriť konfiguračný mapovací skript, ktorý nastavil dátové typy na jednotlivé položky údajov v dokumentoch. Potom mal nastaviť import samotných dát, ale po zdanlivom importovaní databázy bolo vidno, že žiadne údaje sa neimportovali. Bolo to spôsobené tým, že PHP skripty, ktoré importovali údaje do Elasticsearch mali nastavené na import jednotlivých položiek zápis pre formát String. Z tohto dôvodu bol import uskutočnený bez mapovacieho skriptu. Formát v ktorom Elasticsearch prijímal údaje bol JSON a importy prebiehali cez REST API.

3.4 Nastavenie OrientDB

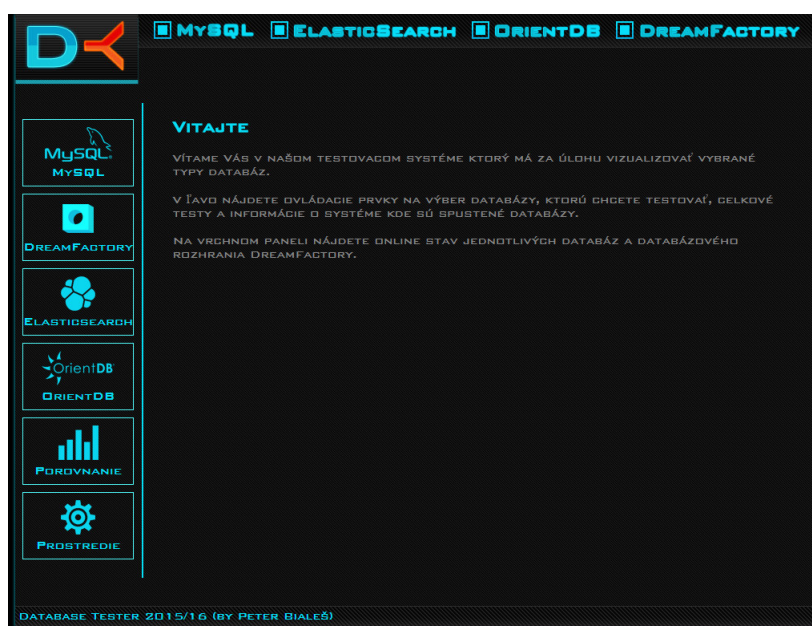
Po importovaní dát do Elasticsearch bolo treba vytvoriť konfiguračné súbory na importovanie údajov do OrientDB. Pred importovaním údajov z MySQL do OrientDB bolo potrebné nakopírovať importovacie konfiguračné súbory do bin priečinka OrientDB. Nasledovnými príkazmi prebiehal import údajov do OrientDB:

```
cd orientdb-community-2.1.9/bin/  
sh oetl.sh 01_cohave_feed.json  
sh oetl.sh 02_funfi_device_score.json
```

Ako vidno z príkazov, každá tabuľka mala svoj importovací JSON súbor.

3.5 Vytváranie Selektov a programovanie webového systému

Po úspešnom importovaní údajov do všetkých databáz nasledovalo vytvorenie špeciálnych selektov pre jednotlivé databázy a vyskúšanie si REST rozhrania v jazyku PHP. Niektoré Selektory budú popísané v nasledujúcej kapitole. Po napísaní selektov nasledovalo programovanie jednoduchého webového systému, ktorý by ukázal na možnosti jednotlivých databáz a vedel by spraviť jednoduchý test rýchlosti vyhľadávania vo všetkých databázach. Náhľad úvodnej obrazovky webového systému je možné vidieť na nasledujúcom obrázku:



Obrázok 1: Náhľad úvodnej obrazovky webového systému

4 Overenie Riešenia

V tejto kapitole budú popísané jednotlivé selekty, ktoré sú špecifické pre jednotlivé databázy a jednotný test, ktorý má za úlohu poukázať na časovú odozvu pri vyhľadávaní v rôznych databázach nad rovnakými údajmi.

4.1 Join selekt pre MySQL

Nasledujúci selekt vyberá z databázy cohave_v3 všetky naskenované wifi siete užívateľa s emailom ferko.mrkvicka@gmail.com a s id zariadenia 338:

```
SELECT ssid FROM general_user u INNER JOIN general_user_device ud INNER JOIN general_device d INNER JOIN funfi_wifi_network n ON (u.id = ud.user_id AND ud.device_id = d.id AND d.id = n.device_id AND ud.device_id = 338 AND email = 'ferko.mrkvicka@gmail.com')
```

Výsledkom je takýto response:

Lenovo A7000-a, apple, sovietky zvaz, abc, PoliklinikaMD, HD_backup_net, kEMMpus, grgnet-hreb2, kEMMpus, kEMMpus, kEMMpus, kEMMpus, Hello122, OnePlus2, gtfo, izba37, Izba36, N@vigat3, APMEDIA-ST, ADB-F0EF53, RH-AP, eduroam, eduroam, orange-free, HP-Print-0F-LaserJet 1102, airlive

Ako vidno zo zápisu, MySQL databáza je špecifická tým, že dokáže spájať tabuľky pomocou Joinov. V tomto prípade bol použitý Inner Join, no existuje ešte Left Join, Right Join, Full Join.

4.2 Vyhľadávací selekt pre Elasticsearch

Elasticsearch je dokumentová vyhľadávacia databáza. Nasledujúci kód má za úlohu vyhľadať na indexe data v typoch dokumntov funfi_wifi_network ssid s názvom apple:

```
GET /data/funfi_wifi_network/_search?q=ssid:apple
```

Ako response dostávame JSON objekt, ktorý obsahuje údaje o vyhľadávani:

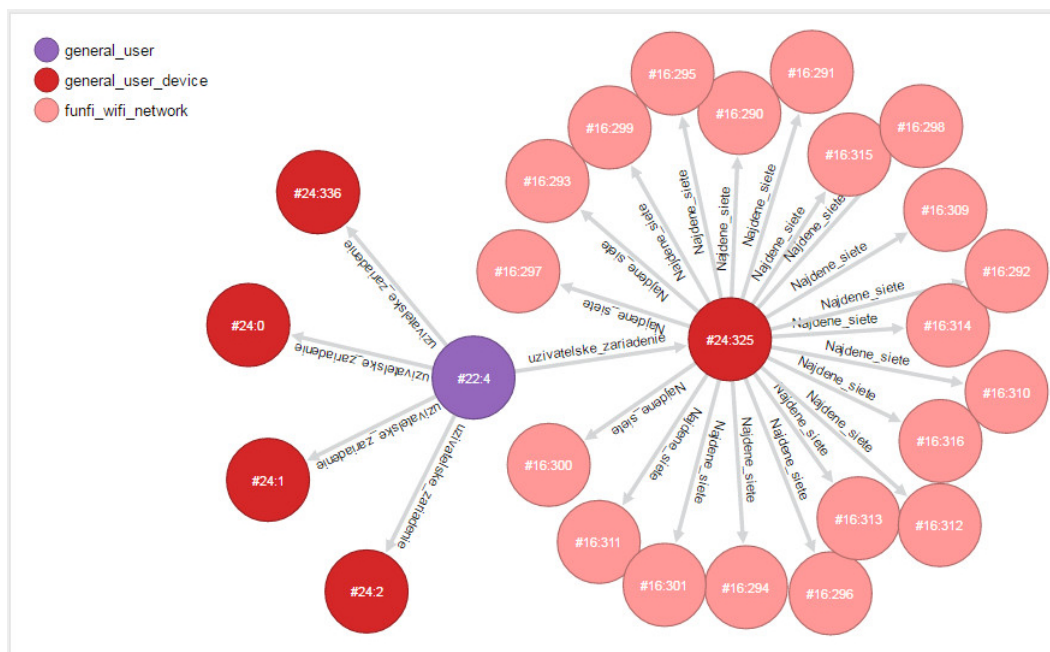
```
{
  "took": 7,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 10.799265,
    "hits": [
      {
        "_index": "data",
        "_type": "funfi_wifi_network",
        "_id": "292",
        "_score": 10.799265,
        "_source": {
          "bssid": "f4:ec:38:fe:24:04",
          "ssid": "apple",
          "frequency": "2462",
          "capabilities":
            "[WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]",
          "device_id": "338",
          "inserted_time": "2015-10-17 10:41:37",
          "app_version": "funfi 1.0.0",
          "log_time": "1445022140"
        }
      }
    ]
  }
}
```


4.3 Selekt pre grafovú databázu v OrientDB

V prípade OrientDB bolo ešte potrebné vytvoriť triedy `najdene_siete` a `uzivatelske_zariadenie`, ktoré boli dedičmi triedy `E`, ktorá reprezentuje v OrientDB hrany. Tieto triedy obsahujú vstupné a výstupné adresy vrcholov týchto hrán. Po naplnení týchto tried údajmi bol zadaný nasledovný selekt do Graph Editora:

```
select from general_user where email = "ferko.mrkvicka@gmail.com"
```

Výsledkom po rozkliknutí zariadení užívateľa a kliknutí na jedného z nich bol tento response:



Obrázok 2: Reprezentácia grafovej databázy v OrientDB

4.4 Zrovnávací test odozvy jednotlivých databáz

Pre testovanie všetkých databáz na rovnaký typ požiadavky bolo potrebné vytvoriť jednotné komunikačné rozhranie a zabezpečiť rovnaké podmienky testov pre všetky typy databáz. Test sa vykonával po reštarte virtuálneho stroja pomocou php skriptov, pričom vyberalo sa z malej tabuľky, stredne veľkej tabuľky a veľkej tabuľky. Dotazy sa vykonávali 10x, 25x a 50x za sebou.

Tieto tabuľky boli použité na testovanie a takáto bola konfigurácia systému pri teste:

	Názov	Počet záznamov	Veľkosť v MB
malá tabuľka	general_device_type	10	0,0024
stredne veľká tabuľka	funfi_wifi_location	57729	5
veľká tabuľka	general_device_log	~111344	1800

Tabuľka 1: Tabuľky použité na testovanie

Konfigurácia systému:

Operačný Systém: Ubuntu 14.04 LTS

Typ OS: 64 bitový

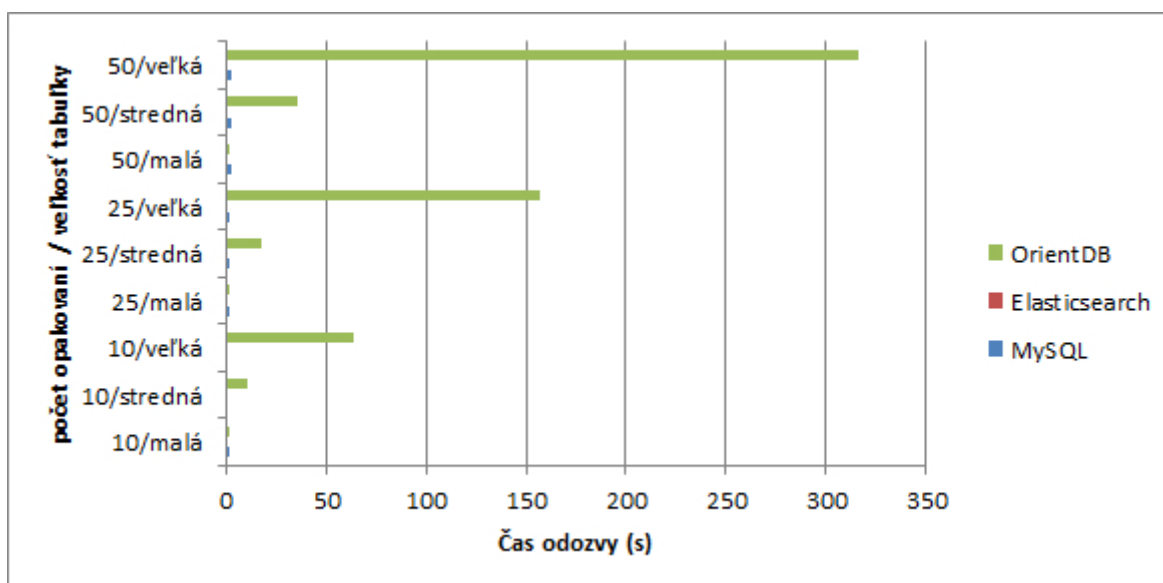
Pamäť: 2,9 GB

Procesor: *Intel®Core™i5 – 3210M CPU@2.50GHz* × 2

Disk: 80,2 GB

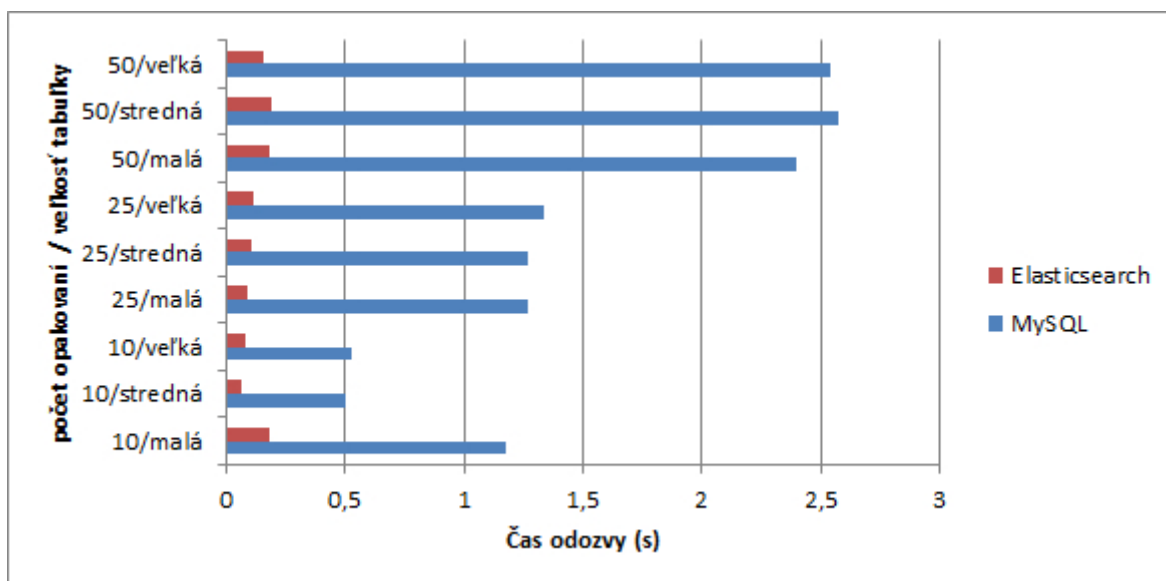
Typ Disku: SSD

Výsledky testovania možno vidieť na nasledovnom grafe:



Obrázok 3: Graf porovnanie odozvy MySQL, Elasticsearch, OrientDB

Pre lepšie porovnanie MySQL s Elasticsearch bol vytvorený nasledujúci graf:



Obrázok 4: Graf porovnanie odozvy MySQL a Elasticsearch

4.5 Zhodnotenie

Ako bolo uvedené, každá databáza je niečím výnimočná. MySQL tým, že je relačná, je dobre optimalizovaná, Elasticsearch tým, že je dokumentová vyhľadávacia databáza a teda záznamy sú uložené v dokumentoch, čo umožňuje aj rýchlejšie vyhľadávanie. OrientDB je výnimočná tým, že je grafová databáza a dajú sa pomocou nej prehľadne interpretovať dáta do grafov.

Z testov vyplynulo, že Elasticsearch mal najmenšiu dobu odozvy vo všetkých prípadoch testovania. Na druhom mieste sa umiestila databáza MySQL s o niečo väčším časom odozvy. OrientDB malo výrazne väčší čas odozvy ako ostatné databázy. Mohlo to byť spôsobené tým, že pri prístupe do OrientDB sa bolo nutné autorizovať.

Záver

V tejto práci sú rozoberané SQL a NoSQL databázy. Práca je zameraná na tri konkrétne typy databáz: MySQL, Elasticsearch a OrientDB. Tieto databázy boli zvolené kôli svojej rozmanitosti a boli podrobne popísané ich výhody a nevýhody a ich využitie. Ďalej je v práci popísaný postup ako sa dajú importovať údaje z relačnej MySQL databázy do Elasticsearch a OrientDB. Implementačná sekcia sa zaoberá Nastavením prostredia operačného systému na ktorom sú databázové služby nainštalované. Popisuje možné úskalia konfigurácie viacerých databázových serverov na jednom operačnom systéme. Po úspešnej konfigurácii a importovaní údajov do databáz sú popisované špecifické selekty pre každú databázu osobitne. V závere práce je popisovaný jednoduchý zrovnávací test všetkých databáz.

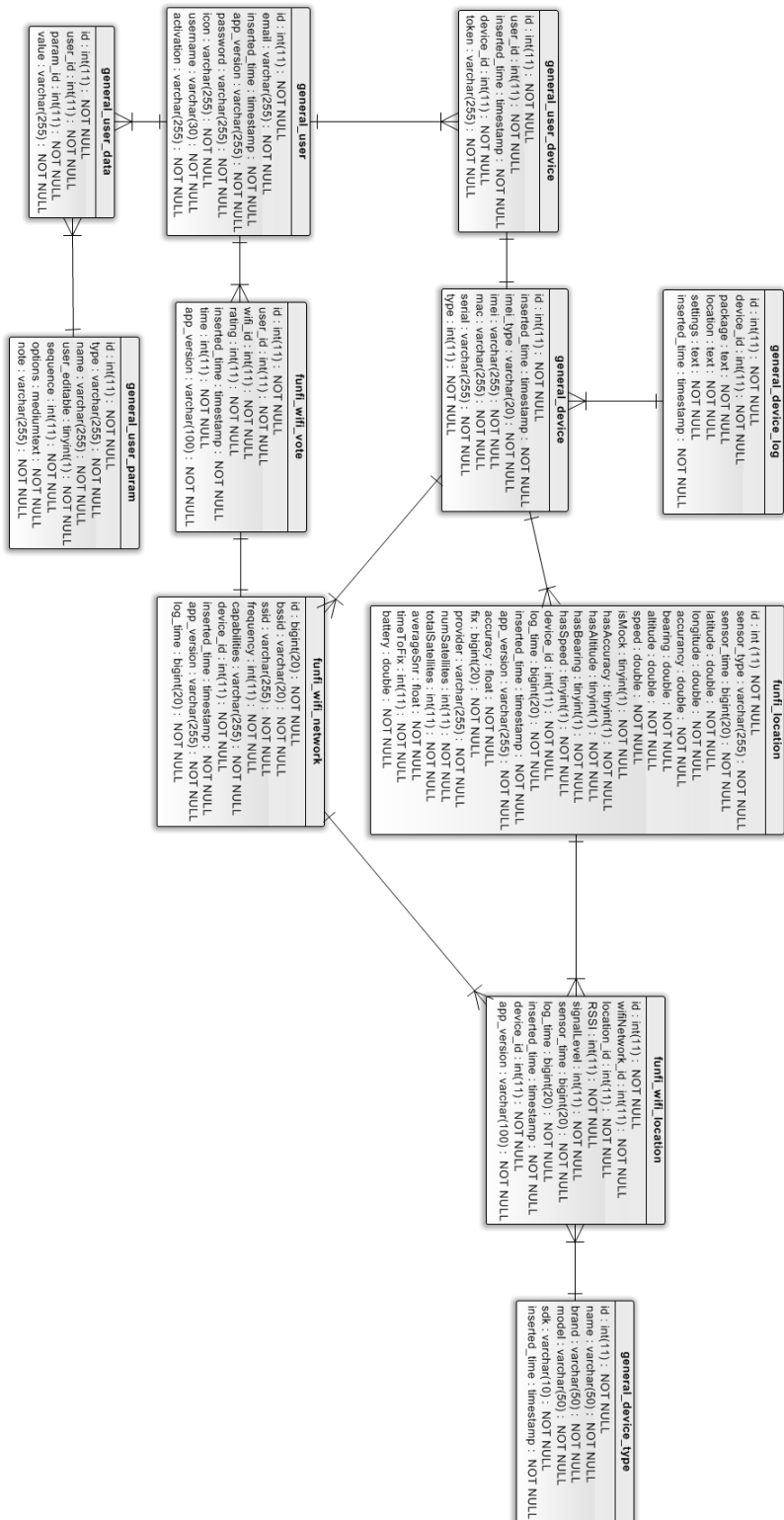
Zoznam použitej literatúry

- [1] ELASTICSEARCH. Basic Concepts | Elasticsearch Reference [2.0] | Elastic, 2016.
- [2] ELASTICSEARCH. Getting Started | Elasticsearch Reference [2.0] | Elastic, 2016.
- [3] ELASTICSEARCH. You Know, for Search... | Elasticsearch: The Definitive Guide [2.x] | Elastic, 2016.
- [4] MICROSOFT. OrientDB Community Edition 2.0.10 on Microsoft Azure Marketplace, 2016.
- [5] ORACLE CORPORATION. MySQL :: MySQL 5.7 Reference Manual :: 1.3.1 What is MySQL?, 2016.
- [6] ORACLE CORPORATION. MySQL :: MySQL 5.7 Reference Manual :: 1.3.2 The Main Features of MySQL, 2016.
- [7] ORIENTDB LTD. Chat | OrientDB Manual.
- [8] ORIENTDB LTD. Time Series | OrientDB Manual.
- [9] ORIENTDB LTD. Why OrientDB? - OrientDB Distributed Graph DatabaseOrientDB Distributed Graph Database, 2016.

Prílohy

A	Testovacia relačná databáza cohavv_v3	II
B	Príklad iportovacieho skriptu pre jednu tabuľku z MySQL do Elasticsearch . . .	III

A Testovacia relačná databáza cohave_v3



B Príklad iportovacieho skriptu pre jednu tabuľku z MySQL do Elasticsearch

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "data";
$table_name="funfi_wifi_score";
$table_labels = array("id", "pos_score", "neg_score", "factor", "updated");

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$pos=0;
$sql = "SELECT ";
while($pos<count($table_labels)-1){
    $sql= $sql. $table_labels[$pos]. ", ";
    $pos++;
}
$sql= $sql. $table_labels[$pos]. " FROM ". $table_name;
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    $counter = 0;
    while($row = $result->fetch_assoc()) {
        $counter++;
        $curl = "curl -XPUT 'localhost:9200/data/" . $table_name;
        $curl= $curl. "/" . $row["id"]. "?pretty' -d '{";
        $pos = 1;
        while($pos < count($table_labels)-1) {
            $curl= $curl. "\". $table_labels[$pos]. "\": \";
            $curl= $curl. $row["$table_labels[$pos]"]. "\"\",";
            $pos++;
        }
        $curl= $curl. "\". $table_labels[$pos]. "\": \";
        $curl= $curl. $row["$table_labels[$pos]"]. "\"}";
        echo exec($curl);
        //echo $curl . "<br><br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```