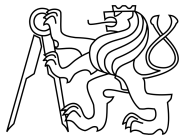
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Implementace prostředí pro experimentování s grafovými databázemi
Student:	Radka Karvánková
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

Navrhněte a implementujte prostředí pro experimentování s grafovými databázemi. Prostředí bude fungovat v linuxové distribuci Debian. Základní požadavky na realizované prostředí jsou:
- nastavení počtu opakování experimentu,
- nastavení vstupu a výstupu jednotlivých experimentů,
- monitoring průběhu (sledování a výpis logů, sledování a zaznamenání času, provedení případně dalších veličin),
- zpracování naměřených hodnot formou tabulek a jednoduchého grafického zobrazení.

Součástí práce je rozbor a volba implementační platformy - skriptovacího jazyka.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 19. září 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Implementace prostředí pro experimentování s grafovými databázemi

Radka Karvánková

Vedoucí práce: Ing. Michal Valenta, Ph.D.

16. května 2017

Poděkování

Velmi děkuji mému vedoucímu práce panu Ing. Michalovi Valentovi, Ph.D. za rady a trpělivost a mé velké rodině za podporu během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 16. května 2017 .....

Odkaz na tuto práci

Karvánková, Radka. *Implementace prostředí pro experimentování s grafovými databázemi*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce se věnuje návrhu a implementaci prostředí pro testování grafových databází. Na základě analýzy je prostředí řešeno sadou skriptů psaných ve skriptovacím programovacím jazyce Python. Výsledek praktické části umožňuje lehce konfigurovat a spouštět jednotlivé experimerimenty nad grafovými databázemi. Prostředí provádí monitorování vykonaných experimentů a umožňuje grafické zobrazení naměřených hodnot. Hlavním přínosem této práce je poskytnutí univerzálního prostředí pro testování a zautomatizované porovnávání různých grafových databází.

Klíčová slova prostředí pro testování grafových databází, prostředí pro spouštění testů, konfigurace testů, monitoring testů, vykreslení grafů v Pythonu, Debian

Abstract

This bachelor thesis deals with the design and implementation of a graph database testing environment. Based on the analysis, the environment is realized by a set of scripts written in Python scripting language. The result of the practical part makes it easy to configure and run individual experiments

over graph databases. The environment performs the monitoring of the executed experiments and allows the graphical display of the measured values. The main benefit of this work is to provide a universal testing environment and automated comparison of different graph databases.

Keywords graph databases, testing of databases, configuration of script, comparison of scripting languages, monitoring of tests, plotting of graph, python

Obsah

Seznam obrázků

Seznam tabulek

Úvod

S příchodem nových technologií přichází i nový pohled na ukládání a správu dat. V některých oblastech vznikají naprosto odlišné požadavky na databázové systémy a tradiční relační databáze nejsou vždy vyhovujícím řešením.

Tento jev má za následek rychlý rozvoj nové generace databází, které se snaží tyto nedostatky pokrýt. V posledních letech vzniklo a stále vzniká mnoho různorodých systémů pro ukládání dat, které popírají dlouhodobě zavedené postupy.

Zajímavou skupinou jsou grafové databáze, které mají velmi odlišný pohled na data. Vznikly především pro potřeby ukládání vztahů mezi jednotlivými objekty například v oblasti sociálních sítí, neuronových sítí, mobilních sítí, vyhledávání vhodných dopravních spojení apod. Získání objektivního přehledu nad rychle se vyvíjejícími grafovými databázemi není lehké.

Výstup této práce pomůže uživatelům, kteří se rozhodují, kterou grafovou databázi použít, s porovnáním měřitelných ukazatelů.

V práci se zabývám analýzou, návrhem a implementací testovacího prostředí pro experimentování s grafovými databázemi. Experimenty, které je možné v tomto prostředí provádět jsou uživatelsky konfigurovatelné a zároveň nad nimi běží monitoring. Uživatel si může do prostředí nainstalovat a následně otestovat libovolnou grafovou databázi. Výsledky experimentů je poté možné zobrazit v podobě jednoduchých grafů a porovnávat.

Tato práce dále pokračuje v následující struktuře. Nejprve se v rychlosti podíváme, co to vlastně jsou grafové databáze. Poté si rozebereme základní požadavky na prostředí. Z toho vyplyne potřeba ujasnit si otázku, jaké technologie na vývoj prostředí použít. Této otázce se věnuji v kapitole Analýza a návrh. A v poslední části práce představím způsob implementace a použití pro uživatele.

Kapitola 1

Cíl práce

V teoretické části práce se budu věnovat specifikaci požadavků na řešení, přehled existujících řešení a rozbor skriptovacích jazyků. Na základě tohoto rozboru provedu výběr vhodného jazyka a jeho knihoven pro implementaci. Bude potřeba vyřešit komunikaci s jednotlivými grafovými databázemi, protože nemají jednotný dotazovací jazyk jako mají databáze relační. Také musím vybrat vhodné uložiště pro ukládání monitorovaných dat.

V praktické části práce se budeme zabývat samotnou realizací. Popíšeme vlastnosti prostředí, způsob implementace a použité metody a jejich výhody a nevýhody.

Kapitola 2

Úvod do grafových databází

V tomto krátkém úvodu popíši postupný vývoj databázových systémů a přiblížím důvody vzniku a hlavní využití grafových databází.

2.1 Vývoj databázových systémů

Tradiční databázové systémy (především relační DBS) jsou stále hlavním způsobem ukládání a spravy dat, ale již několik let nejsou jediným způsobem. Podívejme se společně na vývoj těchto systémů.

2.1.1 Historie databází

Snaha o shromažďování a ukládání dat je na světě od pradávna. Rozvoj informačních technologií měl za následek přenesení této důležité úlohy na výpočetní techniku. Díky tomu začali vznikat v polovině 20. století první koncepce databázových systémů. Postupně byly vyvinuty síťové, hierarchické, relační a objektově-orientované databázové systémy.

Relační databáze prošly mohutným vývojem a staly se nejoblíbenějšími a dosud je tento typ databázového systému nejpoužívanější. Díky relačnímu pohledu na data, tento model splňuje celou řadu požadavků, které v minulosti byly a v současnosti stále ještě jsou kladené na systémy pro ukládání dat. Spolu s relačními databázemi vznikl také dotazovací jazyk SQL, díky kterému se dá velmi efektivně s uloženými daty i s celým systémem pracovat.

2.1.2 Nové potřeby na ukládání dat

Nicméně svět informačních technologií se stále rozvíjí a vznikají stále nové možnosti využití. Spolu s tímto přirozeným jevem vznikají i nové potřeby na ukládání dat. Možnosti současných relačních databází jsou v některých případech příliš omezující a systém se nedokáže vhodně přizpůsobit povaze ukládaných dat. Změny v požadavcích se objevily také s příchodem tzv. Big

data, neboli nutností zpracovávat obrovská množství dat. Například díky senzorovým sítím, sociálním sítím, mobilním aplikacím apod.

ACID transakce pro některé nové problémy ztrácí svou váhu, kde jsou zanedbany pro lepší dostupnost a škálovatelnost systému. Na přelomu millenia byl Ericem Brewerem popsán CAP theorem, který právě tento relativně nový pohled odráží. Podle tohoto theoremu existují zásadní požadavky na systém, přičemž je možné vyhovět maximálně dvěma požadavkům zároveň. Jedné se o Consistency (Konzistence dat), Availability (Dostupnost dat) a Partition tolerance (Tolerance k rozdělení dat).

2.1.3 Vznik moderních databází

V posledním desetiletí vzniklo spoustu nerelačních databázových systémů, které se snaží novým požadavkům vyhovět a vhodně vybalancovat své možnosti pro určitý druh datových úloh. Tyto databáze jsou obvykle nazývané jako NoSQL databáze. Často vznikají podle konkrétních potřeb buď v rukou velkých společností (např. Google a Facebook), nebo v rámci open source komunit. Mají pouze pár společných rysů, protože sem spadá spoustu velmi odlišných uložišť s odlišnými funkcemi a výkonem a tudíž i velmi odlišným využitím. Stejně tak neexistuje pro NoSQL databáze jednotný způsob přístupu k datům, každá NoSQL databáze má svůj vlastní dotazovací jazyk.

Pro hrubou kategorizaci se v literatuře [1] uvádí tyto typy moderních databází:

Databáze typu klíč-hodnota Libovolný typ dat, která jsou uložená na základě unikátního klíče. Data lze vyhledat pouze podle klíče. Velmi rychlé.

Sloupcové databáze Data jsou uložená v tabulce, která nemá definované sloupce. Každý řádek má své vlastní sloupce.

Dokumentové databáze Pro ukládání dat v samopopisných strukturovaných dokumentech, jako je například format JSON nebo XML.

Grafové databáze Určené pro data, která je vhodné modelovat a dotazovat jako grafy. Vhodné pro hledání sousedů, vhodného průchodu grafem apod. Využití v sociálních sítích, aplikacích pro dopravní spojení apod.

My se v této práci zaměříme právě na poslední zmiňovaný typ - grafové databáze.

2.2 Grafové databáze

Grafové databáze představují speciální kategorii NoSQL databází, které se zaměřují převážně na vztahy mezi jednotlivými entitami. Tato data reprezentují jako graf, ve kterém jednotlivé entity představují uzly a vztahy mezi entitami

jsou představují hrany. Uzly i hrany v grafu mají název a mohou mít libovolné atributy, například vzdálenost, délku trvání apod.

U grafových databází určujeme zda jsou hrany orientované či nikoliv. To znamená, zda je vztah obousměrný (např. entity X a Y jsou přáteli), nebo jestli je průchodný pouze s jedné ztrany (např. X je matkou Y). Také rozdělujeme grafy podle možného počtu vztahů mezi dvěma uzly. Jednoduchý graf může mít pouze jednu hranu mezi dvěma uzly, multigraf jich může mít víc. Existuje také pojem hypergraf, který může jednou hranou spojovat 2 a více uzlů naráz.

2.2.1 Způsoby dotazování a manipulace s daty

V nerelačním světě nevládne jeden dotazovací jazyk, jako je SQL pro relační databáze. A dokonce zatím ještě neexistuje jednotný jazyk pro grafové databáze. Každá databáze má svoje vlastní možnosti přístupu k datům. Je i pár univerzálních dotazovacích jazyků, ale nejsou všechny podporovány všemi grafovými databázemi. Shrneme tedy nejběžnější varianty dotazování a manipulace s daty.

```
$ cd titan-1.0.0-hadoop1
$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
09:12:24 INFO  org.apache.tinkerpop.gremlin.hadoop.structure.HadoopGraph
plugin activated: tinkerpop.hadoop
plugin activated: aurelius.titan
gremlin>
```

Obrázek 2.1: Gremlin console [2]

Cypher Hlavní jazyk například pro databázi Neo4j, který je deklarativní podobně jako SQL. Podporuje většinu typů grafových dotazů, které jsou popsané v literatuře. První typ je hledání přímých sousedů (uzlů, s kterými mají nějakou vazbu) popřípadě nepřímých tzv. uzlů v k-sousedství.

Druhý typ dotazu je kontrola dosažitelnosti, uskutečněná hledáním nejkratší cesty mezi uzly. Třetím typem jsou dotazy odpovídající shodám a konkrétně problém isomorfismu podgrafu, protože pouze ten může být vyřešen v konečném čase. Posledním typem jsou shrnutí dotazů, které umožňují určitý druh seskupení a agregace. Cypher není pouze dotazovací jazyk, ale umožňuje i manipulaci s daty jako například UPDATE a DELETE.

Gremlin Gremlin je nízkoúrovňový jazyk pro průchod grafem. Používá velmi kompaktní syntaxi, takže je hůře čitelný. Podle porovnání v článku [3]

je pro databázi Neo4j Gremlin výkonější v porovnání s jazykem Cypher. Na obrázku je vidět Gremlin konzole. 2.1

Blueprints Blueprints je sada rozhraní a implementací pro grafový model. Analogie k JDBC u relačních databází.

2.2.2 Příklady grafových databází

V současné době je nejznámější a nejvíce používanou grafovou databází jednoznačně Neo4j 2.5. Ale je tu spoustu dalších zajímavých projektů například FlockDB od Twitteru, Titan 2.2, OrientDB 2.3, IBM System G, Sparksee, AllegroGraph 2.4 a další.

Obrázek 2.2: Titan

Obrázek 2.3: Orient DB

Obrázek 2.4: AllegroGraph

Obrázek 2.5: Neo4j

Kapitola 3

Požadavky na prostředí pro testování

V předchozí části práce jsme si přiblížili grafové databáze, abychom měli lepší představu o tom, co vlastně budeme testovat v prostředí, které vyvíjíme. Tato kapitola se zaměřuje na popis základních požadavků, které zadavatel na vyvíjené řešení klade.

Požadavky jsou rozdělené na funkční a nefunkční. Záměrem je, aby byly popsány hlavní funkce systému a co nejlépe vymezeny hranice systému. Čili abychom si ujasnili, co systém řeší a co již neřeší.

3.1 Funkční požadavky

Popis funkčních požadavků vznikl na základě sběru dat a následné analýzy. Definuje úkony a procesy, které jsou od řešení očekávány. 3.1

3.1.1 Prostředí umožňuje testovat libovolné grafové databáze

Prostředí musí být univerzální a musí být schopno komunikovat a provádět operace nad jakoukoliv grafovou databází.

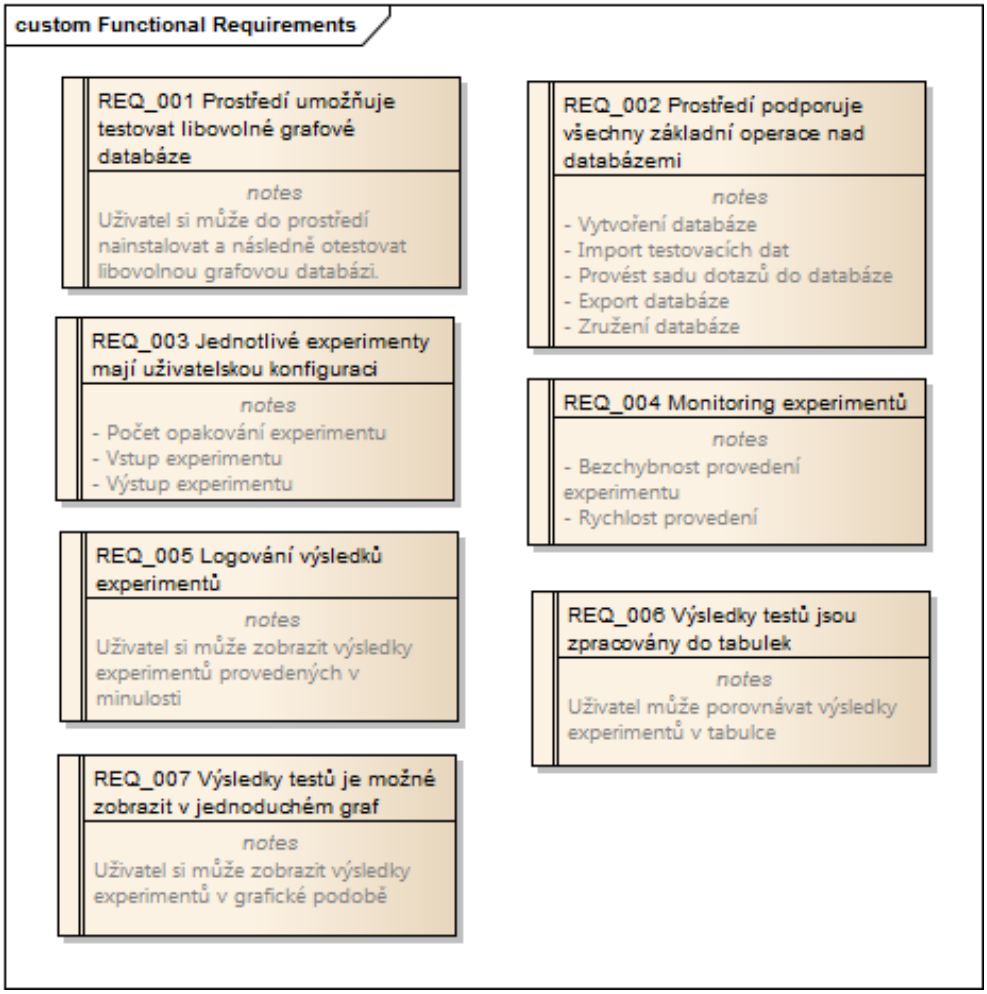
3.1.2 Prostředí podporuje všechny základní operace nad databázemi

Podporou základních operací na databázemi se rozumí, že prostředí umožňuje provádět níže uvedené úlohy:

create Vytvoření databáze

import Import testovacích dat

execute Provést sadu dotazů do databáze



Obrázek 3.1: Přehled funkčních požadavků

export Export databáze

delete Zrušení databáze

3.1.3 Jednotlivé experimenty mají uživatelskou konfiguraci

Experimenty, které budou v testovacím prostředí spouštěny, budou konfigurovatelné a jejich spouštění bude zautomatizované. Experimentu bude možné nastavit tyto parametry:

- Počet opakování experimentu, logovani
- Vstup experimentu
- Očekávaný výstup experimentu

3.1.4 Monitoring experimentů

Experimenty, které budou v prostředí spouštěny, budou monitorovány. U každého experimentu budeme měřit tyto hodnoty:

- Doba běhu experimentu
- Průběžný stav operační paměti během experimentu
- Průběžný stav místa na disktu
- Systémové chyby
- Funkční chyby

3.1.5 Logování výsledků experimentů

Naměřené hodnoty jednotlivých experimentů se budou zaznamenávat a ukládat. Tak aby byly zpětně dohledatelné a porovnatelné s výsledky jiných databází. Pro tyto účely bude vybrán databázový systém, v které budou data uchovávána. Je potřeba navrhnout vhodný datový model.

3.1.6 Výsledky testů jsou zpracovány do tabulek

Je potřeba umožnit zobrazení naměřených hodnot v tabulkových programech, nejlépe vygenerováním CSV souboru.

3.1.7 Výsledky testů je možné zobrazit v jednoduchém grafu

Má být zajištěna vyzualizace naměřených hodnot pomocí jednoduchých sloupcových případně spojnicových grafů. Nejlépe vygenerování PNG souboru.

3.2 Nefunkční požadavky

Následuje seznam doplňkových požadavků kladených na výkonnost, bezpečnost, standardy kvality apod.

3.2.1 Operační systém

Linuxová distribuce Debian

3.2.2 Počet opakování experimentu

Max. 1000

Kapitola 4

Analýza a návrh

Nejprve se podíváme na již existující řešení podobných situací, které budeme řešit my. Poté provedeme rozbor skriptovacích jazyků, na jehož základě vybereme jeden konkrétní jazyk pro implementaci. V závěru se zaměříme na knihovny, které bychom mohli pro specifické úkoly využít.

4.1 Existující řešení

Existuje velké množství knih porovnávajících různé databázové systémy a na internetu jsou také spousty odborných i laických článků o tomto tématu.

Kniha Getting Started with NoSQL [4] například porovnává v různých aspektech několik takzvaných NoSQL(not only SQL) databází.

Také pro zajímavost na webových stránkách DB-Engines rakouské společnosti solid IT můžete vidět měsíčně aktualizovaný žebříček oblíbenosti grafových databází. [5] Oblíbenost se zde vyhodnocuje podle počtu webových stránek, technických diskuzích, profesních profilů a nabídek práce, v kterých je databáze zmíněna. To má celkem vypovídající informaci o velikosti komunity okolo databáze, ale nedozvíme se zda má databáze odpovídající výkon. Viz obrázek 4.1.

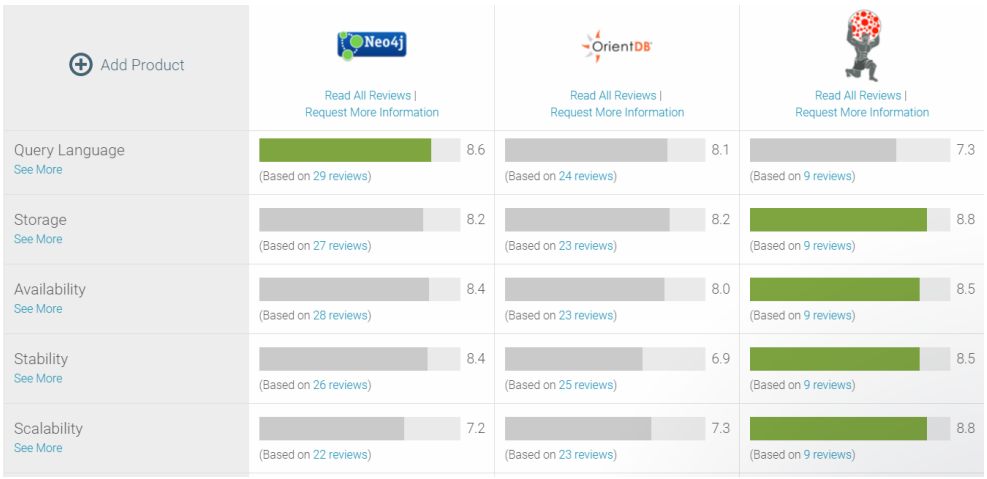
Oproti tomu na webových stránkách společnosti G2 Crowd [6] můžete porovnávat grafové databáze z pohledu hodnocení uživatelů. Dozvíte jak uživatelé hodnotí například použití a administraci databáze, dotazovací jazyk, škálovatelnost, dostupnost a podobně. To může být velmi užitečné, ale né vždy objektivní. Viz obrázek 4.2.

Další zajímavé porovnání databází uskutečnila ve své bakalářské práci Soňa Mastráková [7], která pro otestování různých databází vytvořila v programovacím jazyce Java sedm testovacích nastrojů (tj. jeden nástroj pro každou testovanou databázi). My sice nebudeme dělat jednorázový test, ale i v našem případě by se dalo využít Java rozhraní pro komunikaci s databázemi.

26 systems in ranking, May 2017

Rank May 2017	Rank Apr 2017	Rank May 2016	DBMS	Database Model	Score May 2017	Score Apr 2017	Score May 2016
1.	1.	1.	Neo4j	Graph DBMS	36.15	+1.23	+3.53
2.	2.	2.	OrientDB	Multi-model	5.74	+0.30	-0.39
3.	3.	3.	Titan	Graph DBMS	4.84	+0.21	+0.04
4.	4.	4.	Microsoft Azure Cosmos DB	Multi-model	4.84	+0.56	+2.54
5.	5.	↑ 6.	ArangoDB	Multi-model	2.96	+0.35	+1.39
6.	6.	↓ 5.	Virtuoso	Multi-model	2.06	+0.19	-0.09
7.	7.	7.	Giraph	Graph DBMS	1.08	+0.06	+0.19
8.	8.	↑ 9.	AllegroGraph	Multi-model	0.60	+0.11	+0.10
9.	↑ 10.	↓ 8.	Stardog	Multi-model	0.51	+0.10	-0.04
10.	↓ 9.	↑ 20.	GraphDB	Multi-model	0.50	+0.08	+0.42

Obrázek 4.1: Žebříček grafových databází DB-engines [5]



Obrázek 4.2: Porovnání grafových databází od G2 Crowd [6]

4.2 Skriptovací jazyky

Skriptovací jazyky jsou jazyky interpretované. A to je právě ten hlavní rozdíl od jazyků kompilovaných, které jsou nejprve přeloženy překladačem do strojového kódu jako například C/C++, Pascal nebo Visual Basic. Oproti tomu kód skriptovacích jazyků se nepřekládá předem, ale je přeložen přímo za běhu interpretem. Plynou z toho jisté výhody i nevýhody, které společně s dalšími vlastnostmi skriptovacích jazyků pomáhají vykrýt ta místa, na která kompilované jazyky stačí jen stěží.

Vzhledem k tomu, že se nemusí kód kompilovat, vývoj je rychlejší a díky run-time překladu se skript dokáže dynamicky přizpůsobovat, což předem kompilované programy nedokáží. Na druhou stranu právě kvůli kompilátoru

je běh programů pomalejší a také je kladena o něco vyšší náročnost na paměť. Jsou určeny hlavně pro propojení nebo rozšíření existujících aplikací.

Společné znaky skriptovacích jazyků [8]:

Dávkové i interaktivní využití Řada jazyků dokáže kromě vytváření spustitelných skriptů, provádět příkazy přímo z klávesnice.

Úspornost výrazů Snaha o rychlý, krátký zápis příkazů s co nejmenším použitím interpunkce.

Bez deklarace proměnných Není potřeba deklarovat proměnné. Většina skriptovacích jazykůse drží svých jednoduchých pravidel. Například v Perlu jsou implicitně všechny proměnné globální, v PHP naopak lokální.

Flexibilní dynamické typování V některých jazycích je typ proměnné kontrolován bezprostředně před použitím (Python, Ruby, PHP). V jiných je interpretován v závislosti na kontextu (Perl)

Přístup k systémovému vybavení Komunikace s operačním systémem, manipulace se složkami a soubory, řízení procesů, přístup do databáze, síťová komunikace atd.

Manipulace s textem Obvykle dobře podporují zpracování textů a regulární výrazy.

Složité datové typy Například seznamy a asociativní pole bez potřeby uvolňování paměti.

Skriptovací jazyky se používají například pro:

- Automatizované spouštění programů
- Parametrizaci programů a aplikací
- Web - generování dynamického obsahu
- Správu systému

Skriptovacích jazyků je celá řada od klasických unixových shellů, přes Ruby až po mladičký LiveScript využívaný hlavně pro dynamiku webů. Nyní si některé z nich popíšeme a porovnáme. Pro naše účely nás zajímají hlavně jazyky, které dokáží velmi dobře pracovat se systémem. Do užšího výběru se dostal Bash jako zástupce rodiny shellů, Perl jako klasický skriptovací jazyk, Python jako jeho zdatný nástupce a nakonec moderní Ruby.

4.2.1 Bash

Bash je zkratka pro Bourne Again Shell 4.3. Jedná se o klasický skriptovací jazyk, který vznikl pro automatizované spouštění programů v OS, výpisu textu na obrazovku a práci se soubory v unixových systémech. Podle [9] je shell unixový výraz pro uživatelské rozhraní systému - umožňuje nám komunikovat s počítačem pomocí klávesnice a obrazovky. Ve skutečnosti se výrazem shell označuje interpret příkazů, čili ta část systému, která čte a vykovává příkazy a zároveň vlastní jazyk těchto interpretů [10].

Obrázek 4.3: Bash

Vše začalo u základního Bourne Shellu, který napsal Steven Bourne. Později vznikl C Shell, který se odkazuje podobností příkazů k programovacímu jazyku C, také přidal aliasy a správu více dávkových úloh (job control). A právě spojením Bourne Shellu a C Shellu vznikl Bourne Again Shell, který navíc přidal spoustu dalších vlastností jako například definice funkcí, více řídících struktur, pokročilé řízení vstupu a výstupu atd. Byl vytvořen pro použití v GNU projektu v roce 1988 a rychle se stal standardním shellem zahrnutým v linuxu a široce využit ve UNIX OS a Apple's Mac OSx. Přesto, že tímto vývoj shellů neskončil a vznikly ještě další podobné jazyky (Korn shell, Tenex C shell), bash je stále nejoblíbenější a hojně používaný. [9]

U Bashe bych jednoznačně vyzdvihla práci se souborovým systémem, je skvělý pro zpracování textu a velké množství vestavěných funkcí. Hlavní nevýhodou pak je nepřenositelnost, bash je určen pouze pro unixové systémy. Také nemá jednoznačnou syntaxi, takže se kód může stát hůře čitelný. Není vhodný pro psaní webových stránek (i když to také nějakým způsobem umožňuje), práci s grafickým rozhraním a pro těžší aritmetické operace.

4.2.2 Perl

Jazyk Perl navrhl původně pro svou potřebu systémový programátor Larry Wall v roce 1986. Larrymu nepostačovaly existující prostředky pro zpracování textu a dálkové řízení počítačů. Interpret jazyka byl poskytnut veřejnosti o rok později jako volně dostupný software. Perl je zkratka pro Practical Extraction and Report Language.

Perl je silně inspirovaný jazykem C, unixovým shellem a jinými jazyky. Původně byl navržen pro operační systém UNIX. Postupně byl však jazyk doplňován o další a další vlastnosti. Stal se z něj silný víceúčelový, platformě

nezávislý jazyk, který umožňuje mimo jiné funkcionální, procedurální a objektově orientované programování, multithreading, rozšiřování pomocí modulů a práci s grafickým uživatelským rozhraním. Systémovým administrátorům poskytuje efektivně provádět úkoly zahrnující zpracování textu, práci s databází a síťovou komunikaci. Také je k dispozici překladač Perlu, který vytvoří binární program nevyžadující interpret. [11]

Stinná stránka jazyka Perl je především neexistence pevně definované syntaxe, proměnné se nemusí deklarovat, nelze si definovat vlastní datové typy, neexistuje typová kontrola. Není vhodný pro výuku programování. Filozofie jazyka Perl říká, že správný programátor je líný, netrpělivý a sebevědomý a že existuje více způsobů jak něčeho dosáhnout. Ve skutečnosti umožňuje tato volnost v syntaxi jak velkou efektivitu tak i možnou nečitelnost kódu. Záleží jak si s tímto programátor poradí.

4.2.3 Python

Jazyk Python napsal Guido van Rossum v roce 1990. Narozdíl od jazyka Perl má jasně danou syntaxi a je velmi vhodný pro výuku programování. S tím rozdílem oproti ostatním výukovým jazykům, že se skutečně využívá v praxi.

Má malé jádro a spoustu přípojných modulů. Je jednoduchý, intuitivní a multiplatformí. Umožňuje funkcionální, procedurální i oběktově orientované programování. Skvěle pracuje regulárními výrazy i komplexními čísly. Podporuje práci s CSV a XML, práci se soubory, síťovou komunikaci a vícevláknové programování. Umí vytvářet grafické uživatelské rozhraní a díky frameworku Django je v něm napsáno spousty webových stránek.

Program v jazyce Python může být pomalejší, ale dá se snadno rozšířit pomocí jazyka C, čímž se dají zefektivnit kritické části aplikace.

Podle webových stránek python.cz [12] ve světě Python používají:

- Blender 3D
- Google (YouTube)
- Dropbox
- IBM
- Instagram
- Mozilla
- NASA
- Spotify
- a další, mimo jiné také spousta vědeckých organizací včetně švýcarského CERNu

A jak hlásí tento web, který udržuje rozsáhlá česká komunia okolo jazyka Python:4.4.

Obrázek 4.4: python.cz [12]

4.2.4 Ruby

Jazyk Ruby 4.5 je ze zde popisovaných jazyků nejmladší. Vznikl v roce 1995 jako dynamický, striktně objektový jazyk napsaný v jazyce C. Autorem je Yukihiro Matsumoto, který se silně inspiroval svými oblíbenými jazyky - převážně jazykem Perl. Jeho cílem bylo vytvořit jednoduchý, stručný ale vysoce produktivní jazyk, tak aby bylo programování zábavou. Jazyk Ruby se stal v Japonsku velmi rychle oblíbenějším než jazyk Perl. Celosvětově ho však pořádně proslavil až webová framework Rails. [13]

Přesto, že jazyk nebyl narvžeh pro skriptování a práci se systémem, je možné ho pro tuto práci využít. Také obsahuje knihovnu Shell pro lepší efektivitu práce se systémem. Zvládá skvěle práci s textem, síťovou komunikaci, práci s databází, grafickým GUI apod. Je jednoduchý, intuitivní, dynamicky typovaný a podporuje uzávěry. Platí zde opět jako u jazyka Perl, že jeho syntaktická volnost může způsobit horší čitelnost.

4.2.5 Výběr skriptovacího jazyka

Byl proveden rozbor a jednotlivé skriptovací jazyky byly porovnány viz tabulka 4.1. Při rozboru mě hlavně zajímaly tyto oblasti:

sys Komunikace se systémem

file Zápis a čtení ze souboru

db Knihovny pro komunikaci s databázemi

Obrázek 4.5: Ruby

plot Knihovny pro generování grafů

oop Podpora objektového přístupu

os Přenositelnost mezi OS

syntax Syntaxe, čitelnost, udržovatelnost kódu

docu Dokumentace a dostupnost informací

Tabulka 4.1: Skriptovací jazyky

Skriptovací jazyk	sys	file	db	plot	oop	os	syntax	docu
Bash	+	+	-	+	-	-	-	+
Perl	+	+	+	+	+	+	-	+
Python	+	+	+	+	+	+	+	+
Ruby	-	-	+	+	+	+	+	+

Při výběru vhodného jazyka pro tuto práci byla důležitá především schopnost práce s databázemi, schopnost vykreslit jednoduchý graf, práce se soubory a se systémem. Bash příliš neobstál v komunikaci s databázemi a je určen pouze pro Unixové systémy. Musíme brát v potaz případné rozšíření pro operační systém Windows. Ruby pro změnu není příliš stavěný pro práci se systémem.

Hlavní rozhodování nakonec padlo mezi jazyk Perl 4.6 a jazyk Python 4.7. Oba jazyky mají velmi podobné vlastnosti. Skvělou práci se souborovým

Obrázek 4.6: Perl

systémem, s textem, přístup k databázím. Oba podporují oběktově orientované programování, stejně jako funkcionální a procedurální. Jsou přenositelné, mají dobrou dokumentaci a velkou komunitu okolo sebe. Hlavní rozdíl jsem objevila ve volnosti syntaxe. Které jsem u jazyka Perl označila negativně spíš z toho důvodu, abych vyjádřila rozdílnost v tomto bodě a můj subjektivní názor k volnosti syntaxe.

Každému vývojáři může vyhovovat něco jiného. Dle mého názoru je vhodnější jednoznačná syntaxe, díky, které může být jazyk lépe čitelný a tím pádem i snadněji udržovatelný a předávaný novým členům vývojářského týmu. Ale samozdřejmě i v jazyce s přísnou syntaxí se dá napsat špatně čitelný kód. A naopak i v jazyce s volnější syntaxí se dá napsát krásný čistý kód.

Obrázek 4.7: Python

Dají se mexi jazykem Python a jazykem Perl najít i další rozdíly, ale nejsou takového rázu, že by zásadně ovlivňovaly, nebo znemožňovaly implementovat tuto práci podle popsaného zadání. Nakonec jsem si tedy vybrala jazyk Python. Kromě toho, že k němu chovám větší sympatie, byl jedním z důvodů také fakt, že tento jazyk je u většiny grafových databází podporován. Patří navíc v současnosti mezi nejoblíbenější programovací jazyky, i když uznávám, že to z velké části způsobuje jeho velmi oblíbená webová nadstavba Django a možná také podpora jazyka Python pro psaní IoT(Internet Of Thinks - internet věcí) aplikací.

4.3 Knihovna pro vykreslení grafů

Jazyk Python má k dispozici základní knihovnu pro vykreslení grafů - matplotlib. Skvěle se hodí pro vykreslování složitých vědeckých grafů. Můžete s ní řídit každé písmenko, každý popisek. To ale také znamená, že je složitější na

ovládání. Pokud nejsou nároky na výsledný graf vysoké a očekává se pouze nějaký jednoduchý sloupcový, nebo spojnicový graf, tak jsou vhodné například knihovny Pandas, plot.ly nebo Seaborn.

4.4 Komunikace s grafovými databázemi

Grafové databáze nemají jednotný dotazovací jazyk, takže je potřeba vyřešit jakým způbsobem bude probíhat spouštění dotazů nad databází. A to takovým způsobem, aby se nemuselo zasahovat do nastavení experimentu. Každá databáze bude musí mít svůj vlastní konfigurační soubor. Ke zvážení jsou dvě varianty:

- Konfigurační soubor bude definovat konkrétní operace nad databází a samotnou komunikaci bude zajišťovat skriptovací jazyk.
- Komunikace bude probíhat přes Java rozhraní, které bude implementovat jednotlivé operace nad databází. Skript bude pouze volat rozhraní, nebude vůbec řešit napojení na databázi.

4.5 Databáze pro monitoring

Pro ukládání naměřených hodnot potřebujeme nějakou jednoduchou rychlou open-source relační databázi. Nyní není potřeba žádná robustní databáze, ale je dobré myslet na případné rozšiřování vlastností prostředí. Vybrána byla databáze PostgreSQL pro svou univerzálnost.

Kapitola 5

Realizace

V této kapitole popíši, jaké má testovací prostředí vlastnosti, poté jakým způsobem bylo prostředí implementováno a na závěr jak je možné ho využít.

5.1 Specifikace vlastností

Testovací prostředí pro experimentování s grafovými databázemi je implementováno sadou skriptů v jazyce Python. Jako první je potřeba připravit si prostředí. Poté následuje přípravná fáze, kdy se nastaví konfigurační soubory a poté se spustí samotný experiment. Během průběhu testů probíhá monitoring a naměřené hodnoty jsou zaznamenávány. Následně je možné si naměřená data zobrazit v tabulkovém programu, nebo ve formě grafu.

5.1.1 Instalace prostředí

Pro přípravu prostředí je kromě jiného potřeba nainstalovat databázi PostgreSQL a připravit jí pro ukládání naměřených hodnot z uskutečněných testů.

5.1.2 Přípravná fáze testování

Samotnému spuštění experimentu předchází příprava. Nejprve je nutné nainstalovat grafovou databázi do testovacího prostředí a nastavit konfigurační soubory. Konfigurační soubory jsou dva. Jeden konfigurační soubor se váže k experimentu a nastavují se v něm . 5.1

Tabulka 5.1: Experiment - Konfigurační soubor

název	výchozí	rozsah hodnot	popis	povinné
repetition	1	1-100	Počet opakování experimentu	A
logging	db	db|file|none	Určení místa pro logy	N
querySeq	none	Dotazy	Sekvence dotazů do databáze	N

Druhý konfigurační soubor se váže na testovanou databázi, kde je nutné nastavit způsob komunikace s databází.5.2 Definují se jednotlivé příkazy, které se budou využívat v experimentech.

Tabulka 5.2: Databaze - Konfigurační soubor

název	hodnota	popis	povinná hodnota
dbname	titan	Název databáze	A
create	query	Vytvoření databáze	A
import	query	Import testovacích dat	A
execute	query	Provést sadu dotazů	A
export	query	Export z databáze	A
delete	query	Zrušení databáze	A

5.1.3 Průběh testování

Ve chvíli kdy mám nainstalované prostředí, databázi pro testování a mám upravené oba konfigurační soubory, přichází na řadu spuštění konkrétního experimentu. 5.1

Testování spustíme skriptem runTest.py, kterému můžeme předat několik parametrů. Parametry jsou popsané v tabulce.5.3 Parametr experiment a database jsou povinné, ostatní jsou nepovinné. Pokud je nastaven parametr, který je zároveň nastaven i v konfiguračním souboru, má přednost hodnota parametru uvedená při volání skriptu, nad hodnota z konfiguračního souboru.

Tabulka 5.3: Popis parametrů runTest.py

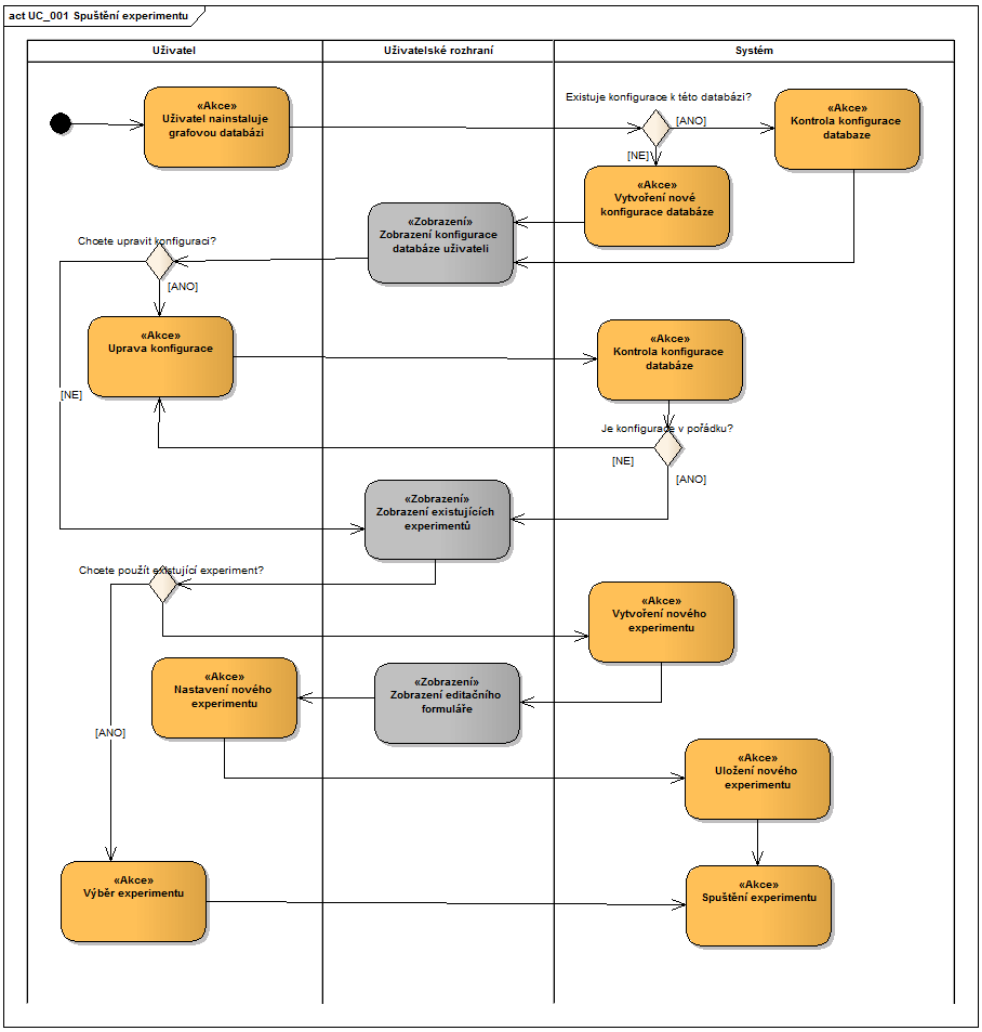
parametr	zkr.	hodnota	popis	povinné
repetition	-r	int	Počet opakování experimentu	N
logging	-l	enum	Určení místa pro logy	N
experiment	-e	file	Konfigurační soubor pro experiment	A
database	-d	file	Konfigurační soubor pro databázi	A
verbose	-v	-	Upovídaný mód	N

5.1.4 Zobrazení a vyhodnocení

Naměřená data se uloží do databáze. Pro zobrazení je několik možností.

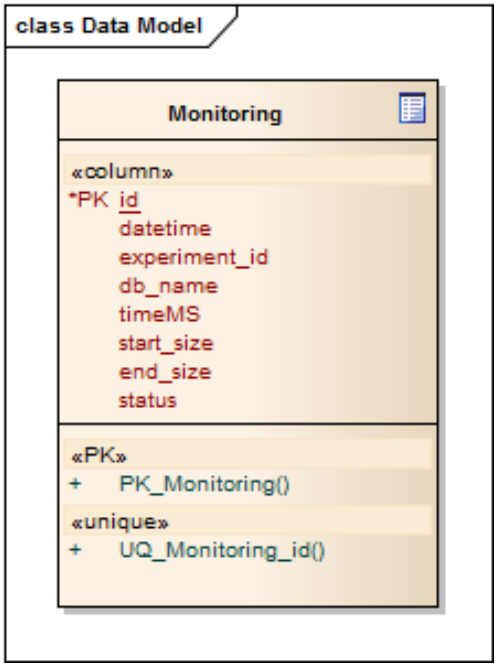
Data se ukládají v databázi GRAPHTEST do tabulky MONITORING.5.2 Popis hodnot ukládaných do tabulky Monitoring:

- id (int) - identifikátor záznamu (primární klíč)
- datetime (Timestamp) - čas spuštění testu



Obrázek 5.1: Diagram aktivit: Spuštění experimentů

- experimentid (int) - identifikátor experimentu
- dbname (varchar(40)) - název testované databáze
- timeMS (int) - doba běhu testu v milisekundách
- startsize (int) - velikost testované databáze před spuštěním testu
- endsize (int) - velikost testované databáze po skončení testu
- status (varchar(40)) - "OK"v případě úspěchu, chybový kód v případě neúspěchu



Obrázek 5.2: tabulka Monitoring

Můžeme se tedy na naměřená data dotazovat přímo do databáze. Například použitím rozhraní psql pro PostgreSQL. Tato varianta je vhodná pro rychlou kontrolu, ale pokud chcete nějaký výstup do souboru, jsou zde k dispozici dvě možnosti.

První možnost je zobrazit si data v tabulce, pro tuto možnost nám slouží skript genData.py, který vybeneruje data z databáze do CSV souboru. Skriptu musíme dát níže uvedené parametry. 5.4

Tabulka 5.4: Popis parametrů genData.py

parametr	zkr.	hodnota	popis	povinné
file	-f	file	Název nového CSV souboru	A
experiment	-e	experimentid	Identifikator experimentu	A
database	-d	dbname	Název databáze	N
verbose	-v	-	Upovídaný mód	N

Druhou možností je zobrazit data ve formě grafu. Skript genGraph.py vygeneruje graf na základě vstupních parametrů popsaných v tabulce 5.5 za použití knihovny Pandas pro jazyk Python.

Tabulka 5.5: Popis parametrů genGraph.py

parametr	zkr.	hodnota	popis	povinné
file	-f	file	Název nového PNG souboru	A
experiment	-e	experimentid	Identifikator experimentu	A
database	-d	dbname	Název databáze	N
verbose	-v	-	Upovídaný mód	N

Závěr

V práci jsem se zabývala analýzou, návrhem a implementací prostředí pro experimentování s grafovými databázemi.

Výsledky mé analýzy poukázaly na možná řešení daných požadavků. Byla provedena analýza knihoven pro grafické zobrazení dat a rozbor skriptovacích jazyků. Byl vybrán vhodný způsob monitoringu a konfigurace experimentů. V praktické části práce jsme testovací prostředí zrealizovali a popsali jeho vlastnosti a způsob implementace. Na závěr jsme zhodnotili výběr technologií a použité metody.

Příjemným rozšířením by mohlo být přidání dalších monitorovaných hodnot a vytvoření uživatelsky přívětivějšího rozhraní. Například umožnit zobrazení výsledků testů (tabulek a grafů) s pomocí webového serveru. Nicméně rozšíření tohoto typu již není předmětem této práce.

Literatura

[1] Kočička, P.; Blažek, F.: *Big Data a NoSQL databáze*. Brno: Computer Press, 2004, ISBN 978-80-247-5466-6.

[2] DataStax: *Titan:db Getting started - dokumentace[online]*. [cit. 2017-05-12]. Dostupné z: `http://s3.thinkaurelius.com/docs/titan/1.0.0/getting-started.html`

[3] Peinl, P. D. R.: *Performance of graph query languages: Comparison of cypher, gremlin and native access in Neo4j*. In: EDBT/ICDT 2013, pp. 195-204. ACM, New York, 2013, doi:10.1145/2457317.2457351.

[4] Vaish, G.: *Getting started with NoSQL: your guide to the world and technology of NoSQL*. Birmingham: Packt, 2013, ISBN 978-1-84969-498-8.

[5] solid IT: *DB-Engines Ranking of Graph DBMS [online]*. [cit. 2017-05-12]. Dostupné z: `https://db-engines.com/en/ranking/graph+dbms`

[6] G2 Crowd: *Best Graph Databases Software [online]*. [cit. 2017-05-12]. Dostupné z: `https://www.g2crowd.com/categories/graph-databases`

[7] Mastráková, S.: *Bakalárska práca: Porovnávacie testy databáz založených na rôznych paradigmách*. Brno, 2013.

[8] Scott, M. L.: *Programming Language Pragmatics*. Morgan Kaufmann, 2009, ISBN 978-0-12-374514-9.

[9] Newham, C.; Rosenblatt, B.: *Learning the Bash Shell : Unix Shell Programming 3rd*. O'Reilly Media, Inc, USA, 2005, ISBN 978-0-596-00965-6.

[10] Libor, F.: *Shell v příkladech aneb aby váš Unix skvěle shell*. Matfyzpress, Praha, 2010, ISBN 978-80-7378-152-1.

[11] Dařena, F.: *Myslíme v jazyku Perl*. Grada Publising, a.s., Praha, 2005, ISBN 80-247-1147-8.

[12] Python.cz: *Python.cz[online].* [cit. 2017-05-12]. Dostupné z: `https://python.cz/`

[13] Hal Fulton, p. J. K.: *Ruby - komendium znalostí pro začátečníky i profesionály.* Zoner software, s.r.o., Brno, 2009, ISBN 978-80-7413-018-2.

Příloha **A**

Seznam použitých zkratek

CSV Comma-separated values - jednoduchý tabulkový formát (data oddělená čárkami)

GUI Graphical user interface - grafické uživatelké rozhraní

IoT Internet of thinks - internet věcí

JSON JavaScript Object Notation - odlehčený datový formát pro výměnu dat

OS Operační systém

PNG Portable Network Graphics - grafický formát určený pro bezeztrátovou kompresi rastrové grafiky

XML Extensible markup language - rozšiřitelný značkovací jazyk vhodný pro výměnu dat

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF