Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Ukládání dat pomocí XML nebo MySQL databáze

Zdeněk Srb

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Ukládání dat pomocí XML nebo MySQL databáze" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 12.3.2015 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Alexandru Vasilenkovi za cenné připomínky a odborné rady, kterými přispěl k vypracování této diplomové práce.

Ukládání dat pomocí XML nebo MySQL databáze

Data storage comparison between XML and MySQL database

Souhrn

Tato práce se snaží přiblížit dvě technologie ukládání dat, relační model a možnost využití značkovacího jazyka XML. Relační model je v práci zastoupen systémem řízení báze dat MySQL a možnosti značkovacího jazyka zastupuje systém řízení báze dat eXist. Oba systémy jsou open source řešením. Pro pochopení základních problematik je nejdříve vysvětlen základní princip ukládání dat při využití obou technologií, následně jsou blíže představeny oba systémy. Stručně je popsán jejich vznik, vývoj a architektura. Dále jsou popsány některé často používané nástroje a postupy pro výkonnostní testování systému řízení báze dat a je definována základní metodika pro testování systémů MySQL a eXist. Na závěr jsou předvedeny výsledky testů, ze kterých jsou vyvozeny závěry práce.

Summary

This thesis introduces two storage technologies, relational model and the possibility of using XML markup language. The relational model in the work represents database management system MySQL and options markup represents the database management system eXist. Both systems are open source solutions. To understand the basic issues is to first explain the basic principles of data storage using both technologies are closer then presented both systems. Briefly described their origin, development and architecture. The following describes some frequently used tools and techniques for benchmarking database management system and is defined by the basic methodology for testing of systems and MySQL eXist. In conclusion, the results of the tests are shown, from which conclusions are drawn work.

Klíčová slova: XML značkovací jazyk, MySQL relační databáze, XPath a XQuery jazyk, XSLT, Data Query Language, metodika testování, srovnávací testy

Keywords: XML markup language, MySQL relational database, XPath and XQuery language, XSLT, Data Query Language, testing methodology, comparative tests

Obsah

1. Úvod

Systémy řízení báze dat tvoří důležitý prvek při manipulaci s datovou základnou, která je základním kamenem většiny webových stránek, služeb, aplikací a podnikových informačních systémů. Tyto systémy v sobě mohou uchovávat různorodá data, ať už se jedná o názory jedinců v podobě databáze uchovávající v sobě data z webových diskuzí a fór, přes počty zákazníků v určitých zemích nebo kontinentech v případě nadnárodních společností, až po lékařské záznamy a osobní data obyvatel celého státu. Během posledních čtyřiceti let se systémy řízení báze dat staly všudypřítomnou součástí informačních technologií. Bylo vyvinuto několik modelů, které definovaly základní předpoklady pro operace s danou datovou základnou. Mezi nejznámější modely patří hierarchický model, síťový model, relační model, dokumentově orientovaný model, objektově orientovaný model a objektově relační model. Tato práce se zaměřuje na problematiku systémů řízení báze dat zastupující relační model a dokumentově orientovaný model. Jako zástupce těchto modelů byly zvoleny následující open source řešení.

- MySQL je open source systém řízení báze dat. Jde o velmi spolehlivý a rychlý systém, který je stále více využíván v oblastech, kde panovali komerční produkty od společností Oracle nebo Microsoft (Dubois, 2014, s. 1,2). Díky různým utilitám obsaženým v balení s MySQL, je správa poměrně snadná. Díky velkému množství rozhraní pro programování aplikací (API), je snadné vytvořit vlastní software pro komunikaci s MySQL.
- eXist si klade za cíl, aby splňovaly požadavky na široké uživatelské základně, tudíž jde o velmi bohatý systém na rozšíření a specifické rysy ve své třídě. Systém řízení báze dat eXist byl navržen v průběhu let a vyvíjen tak, aby uspokojil potřeby uživatelů. Uživatelská základna je velmi pestrá, najdeme zde studenty a profesory na univerzitách, kteří provádějí jazykově zaměřené projekty, nebo velké mezinárodní vydavatele, jenž pracují s miliony dokumentů, až po vývojáře, kteří chtějí rychle vytvářet dokumenty a datově orientované webové aplikace (Siegel, Retter, 2014, s. 10).

Cílem bylo tyto dva systémy řízení báze dat porovnat tím, že bude vytvořena metodika testování, která umožní následné srovnání, i když oba systémy vycházejí z odlišných modelů. Výsledky nemají za úkol najít vítěze a poraženého, ale vyzdvihnout silné a slabé stránky daného řešení.

2. Cíl práce a metodika

Diplomová práce je tematicky zaměřena na problematiku ukládání dat v databázovém systému. Vybrané technologie ukládání dat jsou: relační databázový model, který je pro praktické použití zastoupen systémem řízení báze dat MySQL a dokumentově orientovaný model postavený na základech značkovacího jazyka XML, tuto technologii prakticky zastupuje nativní XML databáze eXist. Hlavním cílem diplomové práce je porovnání kladů a záporů relační databáze a značkovacího jazyka XML. Dílčími cíli jsou:

- Vysvětlení základních principů relačního databázového modelu.
- Analýza možností systému řízení báze dat MySQL, vývojářský tým, vývoj a architekturu.
- Zhodnotit možnosti značkovacího jazyka XML a s ním spojených standardů jako prostředku pro ukládání dat.
- Analýza možností dokumentově orientované nativní XML databáze eXist, a její autorský tým, historii, vývoj a architekturu.
- Zaměřit se na problematiku výkonnostních testů obecně, zhodnocení nejčastějších postupů a možných chyb.

Pro dosažení tohoto cíle je třeba navrhnout postup testování, provést dané srovnávací testy (benchmarky) a z vyhodnocení testů obou metod ukládání dat vyvodit patřičné závěry. Metodika diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Nejdříve budou zhodnoceny základní technologie a standardy využívané na poli systémů řízení báze dat. Představeny budou základní technologie a standardy, které vybrané systémy řízení báze dat využívají. Dále bude osvětlena problematiky výkonnostního testování softwaru obecně a následné praktické ukázky testování databázových systémů. Následovat bude popis některých nástrojů pro vykonávání výkonnostních testů. Hlavního cíle bude dosaženo pomocí návrhu metodiky testování, její provedení a následné srovnání výsledků testů obou metod ukládání dat. Na základě porovnání výsledků z teoretické a analytické části budou formulovány závěry diplomové práce.

3. Teoretická východiska

Následující kapitola se zabývá základními teoretickými východisky relačního modelu systému řízení báze dat a dokumentově orientovaného modelu systému řízení báze dat využívajícího značkovací jazyk XML. Po představení teoretických východisek jsou popsány dva vybrané systémy řízení báze dat, MySQL a eXist, oba tyto systémy mají open source licenci. Stručně jsou zmíněny základní bloky jejich architektury, historie, vývoj a tvůrci. Zde jsou popsány základní teoretické pilíře, o které se opírá analytická část práce.

3.1 Přehled relačního modelu

Databáze je uspořádaná kolekce dat, která je obvykle uložena v jednom nebo více spojených souborech. Data jsou strukturována jako tabulky, mezi kterými je možné dělat křížové odkazy. Pokud existují takové vztahy mezi tabulkami, jedná se o tzv. relační model. Relační model definoval v roce 1970 Dr. Edgar Frank Codd ve článku s názvem *A Relational Model of Data for Large Shared Data Banks*. Dr. Codd později upřesnil svůj model definováním souboru pravidel – tzv. dvanácti Coddových pravidel. Ideální systém řízení báze dat (SŘBD) implementuje všech dvanáct pravidel, toho ale v praxi dosahuje velice malé množství produktů. Výhod relačního modelu je dosaženo i ve výrobcích, které neuplatňují všechna pravidla. MySQL implementuje většinu pravidel, ale některé obchází. Relační model prezentuje data v dvourozměrných tabulkách a dále umožňuje jejich vzájemné kombinace pomocí relační algebry a tím vytvářet pohledy. Schopnost používat tabulky nezávisle nebo ve spojení s jinými tabulkami, aniž by bylo třeba předem definovat jakoukoli hierarchii nebo posloupnost, v níž je třeba data zpřístupnit, činí relační databáze velmi flexibilní (Oppel, 2008, s. 10). Pro návrh a normalizování dobré relační databáze je potřeba znát tři axiomy a dvanáct pravidel.

- Axiom 1: Všechny uložené hodnoty musí být atomické
- Axiom 2: Všechny prvky v relaci jsou jedinečné
- Axiom 3: Žádné další konstrukty

Dr. Codd dokázal, že systém po tyto tři axiomy musí dodržovat následujících dvanáct pravidel.

1. Pravidlo informace: Všechny informace v relační databázi se na logické úrovni reprezentují explicitně hodnotami v tabulkách.

2. Pravidlo zaručeného přístupu: Musí být zajištěno, aby úplně každý údaj v relační databázi byl logicky přístupný použitím názvu tabulky, hodnoty primárního klíče a názvu sloupce.
3. Systematické ošetření prázdných hodnot: Prázdné hodnoty (nikoliv nuly, či prázdné řetězce) jsou systematicky plně podporovány RDBMS pro reprezentaci chybějících informací a neplatných informaci nezávisle na datovém typu. (Typicky řešeno hodnotou NULL).
4. Popis struktury založený na relačním modelu: Popis databáze se na logické úrovni reprezentuje stejně, jako běžná data tzn. v relacích, na které se mohou oprávnění uživatelé dotazovat stejně jako na jakoukoliv jinou relaci.
5. Pravidlo komplexního datového jazyku: Relační systémy mohou podporovat více jazyků a režimů přístupů, ale musí existovat minimálně jeden jazyk, jehož příkazy jsou vyjádřitelné nějakou dobře definovanou syntaxí jako řetězce znaků, který podporuje: definici dat, definici pohledu, manipulaci s daty, omezení integrity, autorizaci. Většina databází využívá jazyk SQL.
6. Aktualizace pohledu: Všechny aktualizovatelné pohledy je možno aktualizovat systémově.
7. Vysokoúrovňová manipulace s daty: Zpracování základní či odvozené relace jako jediný operand se aplikuje jak na vyhledávání, tak vložení a změnu dat.
8. Fyzická datová nezávislost: Aplikace a terminály zůstávají logicky nedotčeny změnami v reprezentaci uložiště nebo přístupových metodách.
9. Logická datová nezávislost: Aplikace a terminály jsou logicky nedotčeny, pokud jsou v tabulkách provedeny změny v uchování informací.
10. Nezávislost integrity: Integritní omezení (viz dále) musí být definovatelné v datovém jazyku v databázi, nikoliv v aplikaci.
11. Distribuční nezávislost: Databázový jazyk musí být schopen manipulovat s daty umístěnými na jiném počítačovém systému.
12. Pravidlo nenarušení: Pokud je v systému více jazyků, žádný z nich nesmí mít možnost manipulovat s daty v rozporu s integritními omezeními.

Prakticky žádný ze současných systémů nesplňuje všechny pravidla naprosto dokonale a spousta programátorů databázových aplikací nevyužívá plně možností relační databáze.

Nejvýznamnějšími výrobci a distributory DBMS jsou firmy Oracle s Oracle Database, IBM s DB2. Dalšími velkými výrobci jsou Sybase a Microsoft s MSSQL.

MySQL, Oracle, Microsoft SQL Server, a IBM DB2 jsou příklady relačních databázových systémů. Takový systém obsahuje programy pro správu relačních databází. Mezi úkoly tohoto systému patří bezpečné ukládání dat, zpracování příkazů při dotazování, analyzování uložených dat a jejich třídění za účelem ukládání nových dat. To vše by mělo být možné nejen na lokálním počítači, ale i v síti. Z tohoto důvodu je databázový systém často označován jako databázový server. Každý program, který je spojen s databázovým systémem, se nazývá databázový klient. Klienti mají za úkol zjednodušit užívání databáze pro koncového uživatele. Uživatel databázového systému většinou není schopen přímo komunikovat s databázovým serverem. Komunikace je příliš abstraktní a velmi nepohodlná. Uživatel očekává přehledné tabulky a listboxy, pro jednoduchou práci s daty. Databázový klient může převzít řadu forem například HTML stránky pro zobrazení a zadávání zpráv v on-line diskuzi, tradiční program s několika okny pro správu adres a schůzek nebo skript v Perlu pro vyřízení administrativních úkolů.

3.2 MySQL

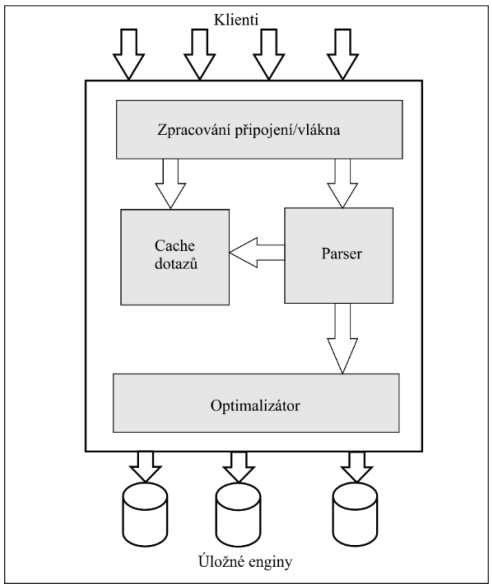
MySQL je open source, vícevláknový a víceuživatelský relační systém řízení báze dat, který vytvořil Michael "Monty" Widenius v roce 1995. V roce 2000 byl MySQL uvolněn v modelu dvojí licence, jenž umožňoval jednotlivcům využívat MySQL jako GNU GPL (všeobecná veřejná licence GNU) zcela zdarma a prominentním klientům možnost zakoupení samotného software nebo know-how řešení. Společnost, která vlastní a vyvíjí MySQL se nazývá MySQL AB (AB je zkratka z aktiabolag, švédský termín pro akciovou společnost). MySQL AB byla 16. ledna 2008 zakoupena společností Sun Microsystems jako akvizice za přibližně 1 miliardu dolarů. 27. ledna 2010 společnost Oracle Corporation získala společnost Sun za přibližně 7 miliard dolarů. I po těchto změnách je stále MySQL distribuován společností Oracle jako open source řešení. Všeobecný úspěch MySQL databázového řešení není jen kvůli jeho ceně, k dispozici jsou i jiné open source řešení, ale také v jeho spolehlivosti, výkonu a vlastnostech. Rychlost je jednou z nejvýznamnějších vlastností. Ve srovnání podle webu eWEEK několika databází včetně MySQL, Oracle, MS SQL, IBM, DB2 a Sybase. MySQL a Oracle měli nejlepší výkon a škálovatelnost. MySQL je pozoruhodně škálovatelné, a je schopno zvládnout desítky tisíc tabulek a miliardy řádků dat. Navíc dokáže řídit menší množství dat velice rychle a hladce.

3.2.1 Licencování MySQL

Ačkoli MySQL lze využívat zdarma a je open source, společnost Oracle drží autorská práva na zdrojový kód. Firma nabízí program pod dvojitou licencí: jeden umožňuje bezplatnost přes GPL za určitých společných podmínek, a druhý je komerční licence požadující poplatky. Oracle umožňuje užívat software zdarma, pokud jej dále nedistribuujete. Další distribuce je možná, pouze pokud cely balíček bude také distribuován pod GLP licencí jako open source řešení. Pokud dojde k vývoji aplikace, která vyžaduje ke své činnosti MySQL a chceme ji dále prodávat, je nutné si zakoupit komerční licenci od společnosti Oracle (11, s. 5). Komerční licence je rozdělena do tří produktů. Tímto MySQL nabízí konečným zákazníkům značnou flexibilitu v konečné volbě. Základní placená verze nese označení „MySQL Standard Edition“, střední řešení „MySQL Enterprise Edition“ a nejvyšší řada „MySQL Cluster Carrier Grade Edition“. „MySQL Community Edition“ je označení pro volně šiřitelnou, nezpoplatněnou verzi. Mimo autorských práv na software, Oracle také vlastní práva na označení MySQL, není proto možné šířit software, který bude v názvu obsahovat tento termín.

3.2.2 Logická architektura

Tato kapitola poskytne přehled o architektuře serveru MySQL, nejdůležitějším je popis architektury úložných enginů (storage engines), jejíž design odděluje vykonávání dotazů a další serverové úlohy. Logický pohled na architekturu MySQL je zobrazen na obrázku 3.1.



Obrázek 3-1 logický pohled na serverovou architekturu MySQL

Nejvyšší vrstva obsahuje služby, nástroje pro klient/server komunikaci. Například obsluha spojení, autentizace, zabezpečení. Druhá vrstva je zajímavější. Zde se nachází valná většina logiky MySQL, včetně kódu pro rozbor dotazu (parser), analýzu, optimalizaci, ukládání do mezipaměti a pro všechny zabudované – „built-in“ funkce (například pro datum a čas, matematické výpočty a šifrování). Na této úrovni se nachází veškerá funkcionalita, která se poskytuje prostřednictvím úložných enginů – triggery, uložené procedury a pohledy. Třetí vrstva obsahuje samotné úložné enginy. Jsou odpovědné za ukládání a načtení všech dat uložených v MySQL. Stejně jako různé souborové systémy, které jsou k dispozici pro GNU/Linux, každý úložný engine má své výhody a nevýhody. Server komunikuje s úložnými enginy prostřednictvím API úložných enginů. Toto rozhraní skrývá rozdíly mezi jednotlivými úložnými enginy a činí je na vrstvě dotazů velmi transparentními (Schwartz, Zaitsev, Tkachenko, 2012, s. 1, 2).

3.2.3 Úložné enginy

MySQL ukládá každou databázi (často označováno také jako schéma) do podadresáře svého datového adresáře na podkladovém souborovém systému. Při vytvoření tabulky, MySQL ukládá definici tabulky do souboru s příponou .frm. Když vytvoříme tabulku s názvem „mojetabulka“, MySQL uloží definici do souboru „mojetabulka.frm“. Rozlišování velikosti písmen je v tomto případě závislé na platformě. Na instanci MySQL na Windows se velikost písmen v názvech tabulek a databází nerozlišuje, na unixových systémech se velikost rozlišuje. Každý úložný engine ukládá data tabulky a indexy jinak, definici tabulky ovšem zpracovává samotný server. Pro zjištění v jakém úložném enginu je určitá tabulka uložena slouží příkaz SHOW TABLE STATUS (nebo od verze 5.0 a výše dotaz INFORMATION_SCHEMA).

InnoDB je standardní transakční úložný engine pro MySQL, nejdůležitější a nejčastěji používaný. Byl navržen pro zpracování mnoha krátkodobých transakcí, které obvykle potvrzují a jen málokdy anulují. Jeho výkon a automatické obnovení po pádu, jej činí populární také pro netransakční potřeby ukládání. Tento úložný engine je používán jako doporučené řešení. InnoDB ukládá data do série jednoho nebo více datových souborů, které jsou souhrnně označovány jako tabulkový prostor (tablespace). Tabulkový prostor je v podstatě černá skříňka, kterou InnoDB spravuje zcela sám. Od verze 4.1 a výše, InnoDB může ukládat data každé tabulky a indexů do samostatných souborů. InnoDB používá řízení souběžného zpracování s více verzemi (MVCC), pro dosáhnutí vysoké souběžnosti a

implementuje všechny čtyři standardní úrovně izolace SQL. InnoDB tabulky jsou postaveny na clusterovém indexu. Důsledkem toho poskytuje velmi rychlé vyhledávání podle primárního klíče. Sekundární indexy (tj. indexy, které nejsou primárním klíčem) obsahují sloupce primárního klíče, takže pokud je primární klíč velký, ostatní indexy budou rovněž velké. Pokud tedy je nad tabulkou hodně indexů, měla by být snaha o co možná nejmenší primární klíč. InnoDB má řadu vnitřních optimalizací. Patří mezi ně prediktivní čtení napřed pro předčerpávání dat z disku, adaptivní hash index, který automaticky vytvoří hash indexy v paměti pro rychlé vyhledávání a insert buffer pro urychlení vkládání. Jako transakční úložný engine, InnoDB podporuje on-line zálohování pomocí různých mechanismů, včetně Oracle proprietary MySQL Enterprise Backup a open source Percona XtraBackup. Ostatní úložné enginy nepodporují tuto formu zálohování, pro získání konzistentní zálohy je nutné zastavit všechny zápisy do tabulky, což v případě kombinace čtení/zápis obvykle skončí zastavením i čtení (Schwartz, Zaitsev, Tkachenko, 2012, s. 16, 17).

MyISAM byl výchozí úložný engine ve verzích 5.1 a starší. MyISAM poskytuje fulltextové indexování, komprese a prostorové funkce (GIS). Nepodporuje transakce ani zámky na úrovni řádků. Jeho největší slabinou je nulová ochrana proti náhodným pádům databáze. Pokud je třeba uložit data pouze pro čtení, nebo pokud tabulky nejsou velké, je možné využít tento úložný engine. MyISAM ukládá každou tabulku ve dvou souborech: datového souboru a soubor indexů. Soubory mají koncovky .MYD a .MYI. Tabulky mohou obsahovat buď dynamické, nebo statické (s pevnou délkou) řádky. MySQL automaticky určuje, jaký formát použít na základě definice tabulky. Počet řádků MyISAM tabulce je omezen především podle volného místa na databázovém serveru. Vzhledem ke kompaktnímu ukládání dat a jednodušší konstrukci, MyISAM může poskytnout dobré vlastnosti pro specifická použití (Schwartz, Zaitsev, Tkachenko, 2012, s. 17, 18).

Systém řízení báze dat MySQL nabízí velké množství ostatních úložných enginů, které je možné použít pro spravování báze dat. Tyto enginy mají různé vlastnosti a rozšiřují tak možnosti použití systému MySQL, jejich popis však není náplní této práce.

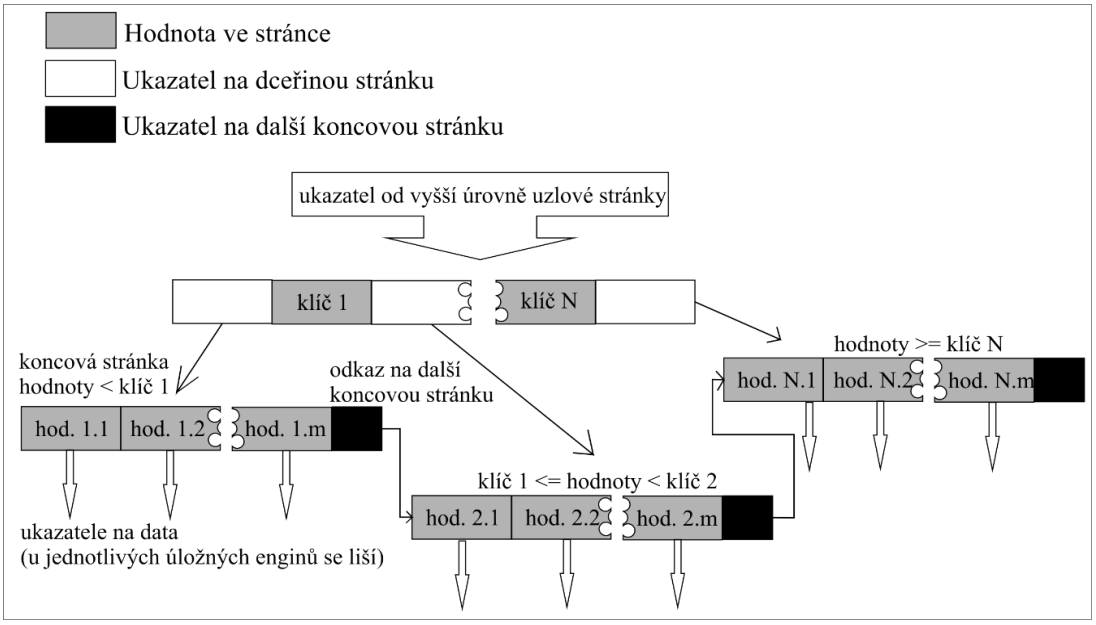
3.2.4 Indexování v MySQL

Indexy (nazývané také "klíče" v MySQL) jsou datové struktury, které úložné enginy používají k rychlejšímu vyhledávání požadovaných dat. Index v databázovém systému funguje podobně jako v knize. Pro rychlé vyhledání určitého termínu, který je diskutován

v knize, vyhledáme jeho zmínky v indexu knihy. Systémy řízení báze dat používají indexy stejným způsobem. V MySQL je dostupnost indexů a jejich typů vázaná na typ použitého úložného enginu. Index obsahuje hodnoty z jednoho nebo více sloupců v tabulce. Pokud indexujeme více než jeden sloupec, pořadí sloupců se stává velmi důležité, protože MySQL účinně vyhledává podle levé krajní předpony indexu. Vytvoření indexu na dvou sloupcích není stejně jako vytvoření dvou samostatných indexů nad jedním sloupcem. Existuje mnoho typů indexů, které byly navrženy tak, aby správně fungovaly pro různé účely. Indexy jsou realizovány ve vrstvě úložných enginů, tudíž nejsou standardizovány.

B-Tree indexy

Nejpoužívanější index v MySQL systému, který podporuje většina úložných enginů. Obecný postup B-Tree indexů spočívá v tom, že všechny hodnoty jsou uloženy v pořadí, a že každá koncová stránka (list stromu) je ve stejné vzdálenosti od kořene stromu. Abstraktní reprezentaci indexu je možné vidět na obrázku 3-2. Index urychluje přístup k datům, protože úložný engine nemusí procházet celou tabulku, pokud má najít požadovaná data. Místo toho začne na kořenovém uzlu. Sloty na kořenovém uzlu obsahují ukazatele (pointers) na dceřiné uzly a úložný engine postupuje podle těchto ukazatelů. Správný ukazatel hledá tak, že prochází hodnoty ve stránkách uzlu, který definuje horní a dolní mez hodnot v dceřiných uzlech. Engine nakonec určí, že požadovaná hodnota neexistuje, nebo úspěšně dosáhne koncové stránky (leaf page) (Schwartz, Zaitsev, Tkachenko, 2012, s. 149).



Obrázek 3-2 B-Tree index struktura (Schwartz, Zaitsev, Tkachenko, 2012, s. 149)

Koncové stránky jsou zvláštní, protože obsahují ukazatele na indexovaná data namísto odkazy na další stránky. (Různé skladování motory mají různé druhy "ukazatelů" na data.) Obrázek 3-2 znázorňuje pouze jeden uzel stránky a její listové stránky, většinou je více úrovní uzlů stránek mezi kořenem a listy. Hloubka stromu závisí na tom, jak velká je tabulka. Jelikož B-Tree index ukládá indexované sloupce v určitém pořadí, značně tím urychluje následné vyhledávání rozsahů uložených dat, urychluje i rovnání výsledných hodnot pomocí příkazu ORDER BY. Oproti tomu B-Tree index trpí několika nedostatky, není příliš účinný, pokud vyhledávání nezačíná z levé strany indexovaného sloupce a nedokáže optimalizovat vyhledávání při použití podmínky rozsahu (LIKE) v dotazu.

Hash index

Hash index je postaven na hash tabulce a je vhodný pouze pro přesné vyhledávání, které používá všechny sloupce s indexem. Pro každý řádek úložný engine vypočítá hash kód indexovaného sloupce. Hash kódy jsou uloženy v indexu a ukazatel na každý řádek v hash tabulce. V MySQL pouze Memory úložný engine podporuje hash indexy. Jde o výchozí typ indexu pro tabulky uložené v tomto enginu, index B-Tree je podporován také. Memory engine podporuje neunikátní hash indexy. Pokud více hodnot má stejný hash kód, bude index uchovávat jejich ukazatele ve stejné položce hash tabulky pomocí propojeného seznamu. Díky tomu, že index obsahuje jen krátkou hash hodnotu kódu, indexy je velmi kompaktní a díky tomu jsou dotazy, které dokáží využít tento index velice rychlé. Zde je několik omezení, se kterými je třeba počítat při užití tohoto indexu. Tento index nelze využít pro dotazy, kde je třeba třídit data do určitého pořadí, index neudržuje řádky v setříděném pořadí. Nepodporuje shodu s částí klíče, protože hash se počítá z celé indexované hodnoty. Pokud tedy máme index na (A, B) a v dotazu v klauzuli WHERE se dotazujeme na A index nám nepomůže. Index podporuje pouze porovnávání na rovnost (operátory =, IN(), <=>) a nemá žádný vliv na rozsahové dotazy (WHERE hodnota > 10). Tyto omezení způsobují, že hash indexy jsou užitečné jen ve specifických případech. Když ovšem vyhovují potřebám aplikace, dramaticky dokáží zvýšit výkon.

Prostorové (R-Tree) indexy

MyISAM podporuje prostorové indexy, které je možné používat s datovými typy pro prostorová geografická měření. Na rozdíl od B-Tree indexů nepožadují prostorové indexy, aby se v klauzuli WHERE operovalo s nejvíce levým prefixem indexu. Dochází k indexaci

dat podle všech dimenzí současně. Důsledkem je, že při vyhledávání se dá efektivně používat jakákoliv kombinace dimenzí. Aby toto fungovalo, je nutné používat speciální funkce GIS MySQL (např. MBRCONTAINS()). ((Schwartz, Zaitsev, Tkachenko, 2012, s. 157)

Fulltextové indexy

Fulltext je speciální typ indexu, který najde klíčová slova v textu místo porovnávání hodnot proti hodnotám indexu. Fulltextové vyhledávání je zcela odlišné od ostatních druhů vyhledávání shody. Využívá mechanik, jako jsou stopslova, kmeny slov, množná čísla a booleovské vyhledávání. Je mnohem více analogické s webovým vyhledávačem. Fulltextové indexy jsou určené pro operace MATCH AGAINST, nikoliv pro obyčejné klauzule WHERE.

3.2.5 Shrnutí

MySQL má vrstvenou architekturu, se serverovými službami a vykonáváním dotazů v horní vrstvě a úložné enginy pod ní. API úložných enginů je nejdůležitější. Spolupráce těchto dvou vrstev je základem serverové architektury MySQL. MySQL byl navržen a postaven kolem enginu ISAM (později MyISAM). Ostatní úložné enginy a transakce byly přidány později. Například způsob jakým MySQL provádí transakce při zadání příkazu ALTER TABLE je přímým důsledkem architektury úložných enginů, stejně jako skutečnost, že datový slovník je uložen v .frm souborech. Velké množství úložných enginů má i své stinné stránky. Někdy volba není dobrá věc, a exploze úložných enginů ve dnech verze MySQL 5.0 a 5.1 zavedly příliš mnoho možností výběru. Nakonec se InnoDB ukázal jako správná volba pro přibližně 95% uživatelů. Ostatní úložné enginy jsou připraveny pro speciální případy a alternativy. Oracle akvizici nejprve InnoDB a pak MySQL přivedl oba produkty pod jednou střechu, kde mohou být dále vyvíjeny. InnoDB engine se zlepšuje mílovými kroky a MySQL zůstává pod GPL licencí a plně open source. Zákazník získá stabilní databázi a server je stále více rozšiřitelný a užitečný.

3.3 Charakteristika XML

XML (Extensible Markup Language) vychází ze základů jazyka SGML (Standard Generalized Markup Language), který byl promyšlený a schopný definovat vlastní značkovací jazyk, ale byl značně složitý. Proto v roce 1990 začala práce na podmnožině jazyka SGML, která dostala označení XML. První pracovní návrh byl publikován v roce 1996 a o dva roky později (10. 2. 1998) konsorcium W3C zveřejnilo revidovanou verzi.

XML byl koncipován jako řešení problému předávání dat mezi různými komponenty. Tuto činnost udělal mnohem jednodušší a zjednodušil starost o různé formáty vstupu a výstupu. XML také zjednodušuje čtení dat jak pro lidi, tak i pro software. XML také umožňoval kompletní oddělení dat a jejich prezentace. To znamená, že stejná data mohou být použita v různých prezentacích a při přenosu těchto dat není třeba informací o jejich vzhledu. V jazyku XML nejsou k dispozici žádné vestavěné prezentační funkce na rozdíl od HTML.

XML dokument je tvořen kolekcí textových dat. Oproti textovému souboru jsou data v XML dokumentu popsána pomocí předem definovaného systému značek, které umožňují zachytit i složité vnořené struktury a díky nimž jsou takto vytvořené dokumenty samopopisné. Tímto lze považovat XML dokument za databázi. Pro popis datového schéma lze využít jazyky DTD (Document Type Definition) nebo XML Schema, funkci jazyka pro definici dat plní díky samopopisnosti sám XML dokument, provádět dotazy nad dokumenty je možné s pomocí jazyka XPath nebo XQuery. Data jsou uložena přímo v souborovém systému bez dalších optimalizací. Nelze využívat indexy a transakce, přístup více uživatelů a bezpečnost nad rámec běžného souborového systému.

3.3.1 Typy XML dokumentů

Dokumenty XML z hlediska jejich obsahu dělíme na dokumentově a datově orientované (document – centric a data – centric). Tyto typy mají odlišné nároky na zpracování.

Dokumentově orientované XML se používá k publikování a různým způsobům opětovného využití obsahu. Struktura dokumentu bývá nepravidelná, významnou část tvoří elementy se smíšeným obsahem, který je tvořen kombinací textu a vnořených elementů (nejmenší nezávislá jednotka dat může být na úrovni prvku s kombinovaným obsahem, nebo celého dokumentu). Pořadí elementů je většinou významné. Obsah těchto dokumentů je většinou psán ručně přímo v XML nebo je získán konverzí z formátů RTF, PDF, SGML do XML a je primárně určen pro koncové zpracování člověkem.

Datově orientované dokumenty jsou dokumenty, které používají XML pro přenos dat. Jsou primárně určeny pro koncové zpracování strojem. Dokumenty se vyznačují pravidelnou strukturou (nejmenší nezávislá jednotka dat je na úrovni PCDATA pouze prvek nebo atribut), malý nebo žádný smíšený obsah. Příkladem mohou být textové reprezentace

relačních dat z databáze, informace z finančních transakcí a datové struktury programovacích jazyků. Příklady obou typů XML dokumentů jsou v příloze číslo 1 a 2.

3.4 XML technologie

V této kapitole budou zmíněny základní technologie a standardy využívané ve světě XML. Zde zmíněné technologie a standardy umožňují využívání XML v řadě případů např.: databázový systém, dokumentový managment, webové služby a reprezentace obrazu.

3.4.1 Parser

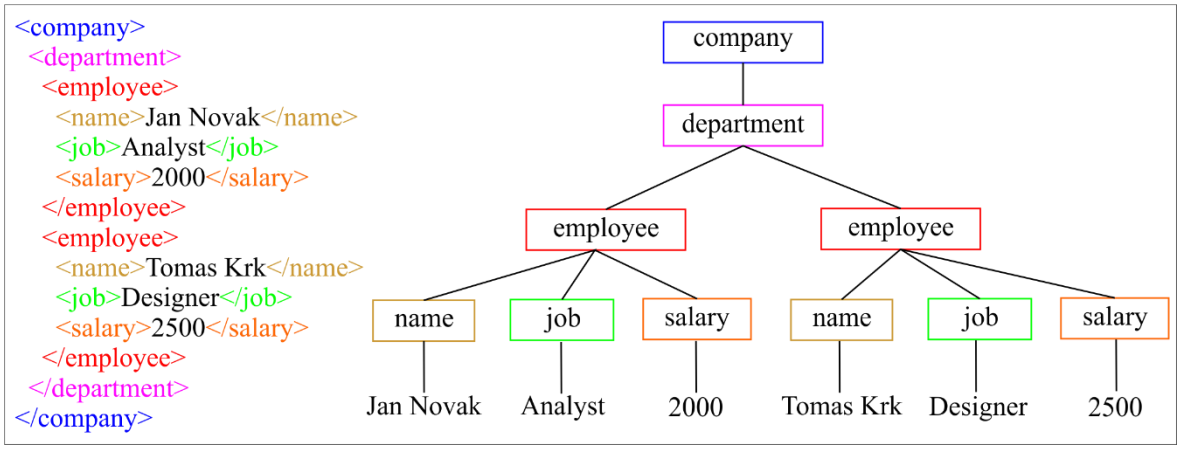
Předtím než můžeme provést jakoukoliv operaci s XML dokumentem je třeba jej parsovat. To znamená zkontrolovat syntaktickou správnost proti nějakému druhu interního modelu. Tímto modelem je nejčastěji DTD nebo XML schéma. Většina parserů se dá spustit z příkazové řádky a jako parametr se jim předá dokument, který mají zkontrolovat. Parser je samozřejmě zabudován i ve všech prohlížečích s podporou XML. Parserů existuje velké množství, komerčních i volně dostupných. Nejčastěji používané jsou tyto:

- MSXML (Microsoft Core XML Services): Standardní balíček nástrojů pro práci s XML od společnosti Microsoft obsahuje i parser. Je dostupný jako COM objekt, je možné jej využít v libovolném jazyce (C, C++, Perl, Java).
- Saxon: Nabízí nástroje pro parsování, transformování a dotazování nad XML daty. Saxon nabízí verzi pro Javu a .NET, základní verze je zdarma.
- Java bulit-in parser: Knihovna Javy má vlastní parser, jeho základní funkce jsou vhodné pro XML úkoly. Knihovnu je možné nahradit za externí implementaci jako je Xerces nebo Saxon.
- Xerxes: Xerces je implementace od open source Apache Software Foundation. Používá se jako základ pro mnoho Java aplikací založených na XML.

3.4.2 Document Object Model

Document Object Model (DOM) je XML aplikační rozhraní standardizované organizací W3C. První verze (DOM Level 1) pochází stejně jako XML 1.0 z roku 1998. Poslední verze je zatím Level 4 s podporou HTML 5. DOM poskytuje standardní sadu objektů pro reprezentování HTML a XML dokumentů, standardní model pro spojení objektů a standardní rozhraní pro přístup a manipulaci s nimi. DOM je stromová hierarchická reprezentace XML dokumentu, kdy každému elementu odpovídá jeden uzel stromu.

Rozhraní DOM obsahuje funkce, které nám umožňují celý strom dokument procházet, modifikovat jeho jednotlivé uzly, mazat je a přidávat. Není nutné procházet dokument od začátku do konce, ale je možné se v něm pohybovat dle potřeby, tato vlastnost se uplatní v aplikacích, které provádějí náročné operace s dokumenty – editory, prohlížeče, kde nachází časté uplatnění ve skriptovacích knihovnách (jQuery). Celou stromovou strukturu je během práce nutné držet v paměti, vzhledem k velkému množství informací, které DOM strom zachovává, se toto může jevit problematické při práci s velkými objemy dat. Struktura DOM a fragment XML dokumentu jsou naznačeny na obrázku 3-3 (Fawcett, Quin, Ayers, 2012, s. 211).



Obrázek 3-3 struktura DOM

Aplikační rozhraní je objektově orientované a nezávislé na programovacím jazyku. Existují knihovny pro implementaci do většiny běžných programovacích jazyků.

3.4.3 SAX

SAX (Simple API for XML) je založen na řízení pomocí událostí (event-driven). Rozhraní nám umožňuje definovat funkce, které se zavolají v okamžiku, kdy parser narazí na začátek elementu, na obsah elementu, na konec elementu, na komentář na instrukce pro zpracování apod. Naší funkci jsou pak předány všechny potřebné parametry jako např. název elementu. Výhoda tohoto přístupu je v jeho rychlosti a malé spotřebě paměti. Jednotlivé události jsou vyvolávány postupně, jak je dokument parsován. SAX na rozdíl od rozhraní DOM nevyžaduje načtení celého dokumentu předtím, než s ním začneme pracovat. Nevýhoda je, že se nelze v proudu dat zpětně vracet. Pokud tedy nepotřebujeme funkčnost DOMu, vyplatí se použít SAX, protože naše aplikace bude rychlejší a bude mít menší

paměťové nároky. Rozhraní není definováno pomocí žádného standardu konsorcia W3C nebo jiné standardizační organizace.

3.4.4 Document Type Definitions a XML Schema

Document Type Definitions (DTD) a XML Schema slouží k popisu definice XML dokumentu, jeho struktury a jaká data jsou v něm povolena, definujeme tím elementy a atributy, které budou k dispozici a jak je půjde navzájem používat. Ty jsou poté použity k testování, zda je dokument v souladu s předepsaným formátem, proces známý jako validace. DTD je starší standard a byly využíván již v SGML. Výhody přináší použití již existujícího DTD. Pokud pro dokument použijeme DTD, které se běžně užívá, získáme tím mnoho jednoúčelových nástrojů, které práci s dokumenty usnadní. XML schémata nabízí více funkcí a jsou napsána v XML. Hlavní nevýhody DTD jsou slabá typová kontrola a nestandardní syntaxe, obě tyto slabiny pocházejí z jazyku SGML. Kromě ověřování dokumentu se DTD a XML Schema také používá k nastavení autorství XML dokumentů. Editory umožňují vytvořit dokument na základě specifických parametrů schématu, během editace nabízejí platný výběr ze schématu a také uživatele upozorní, pokud použili prvek nebo atribut na špatném místě. Většina nových XML formátů je popsána schématy než za pomoci DTD.

3.4.5 XSLT

Extensible Stylesheet Language Transformace (XSLT) umožňuje převod dat z jednoho formátu do druhého. K tomuto převodu je využíván XSLT procesor. Procesorem je míněn program podporující tuto transformaci, tento program může být napsán v libovolném jazyku nebo využít knihovny XSLT daného programovacího jazyka. První verze převáděli jen XML dokumenty na textový výstup, od verze 2.0 je možné použít jako vstup jakýkoliv textový dokument. XSLT je deklarativní jazyk a používá šablony na definování výstupu. XSLT je nejčastěji používán k transformaci XML do XHTML jak na straně serveru tak i klienta. Standard XSLT verze 2.0 byla přijata jako standard konsorcia W3C v roce 2007. Dnes je k dispozici celá řada softwarových procesorů, které umožňují zpracování XML dokumentů na základě XSLT stylu. Většina z nich je přitom napsaná v Javě, některé jsou k dispozici i v C++. Mezi volně šířené procesory patří XT, Saxon a Xalan (Fawcett, Quin, Ayers, 2012, s. 19).

3.4.6 Xpath

Jazyk XPath (XML Path Language) umožňuje najít specifický element nebo atribut uvnitř XML dokumentu. Jazyk je standardizován organizací W3C a je používán v mnoha dalších doporučeních, jako jsou XML Schema, XSLT nebo jazyk XPointer. Z druhé verze jazyka XPath vychází i pokročilejší dotazovací jazyk pro XML dokumenty XQuery. Nejčastěji je předán XPath výraz a jeden nebo více XML dokumentů XPath enginu. Engine vyhodnotí výraz a vrátí zpět výsledek. Toto může probíhat přes API programovacího jazyka, pomocí samostatné příkazového řádku programu nebo nepřímo, když je XPath zahrnut v jiném jazyce (např.: XQuery nebo XSLT). Výsledek vyhodnocení výrazu XPath je seznam uzlů, v praxi je to nejčastěji DOM stromová struktura. V novějších verzích je možné získat zpět XDM sekvenci, ale záleží na použitém API a prostředí jazyka.

Nejdůležitější konstrukcí XPath je cesta k XML uzlu (Location Path). Tato cesta je složena z několika kroků, každý krok se skládá ze tří komponent: identifikátoru osy, testu uzlu (ten je povinný) a podmínky (predikátu). Kroky se spojují lomítky a vyhodnocují se zleva doprava. Při vyhodnocení se vychází z předchozího výsledku a na začátku vycházíme z aktuálního uzlu. Výsledkem může být libovolný uzel, ať už jde o element, atribut nebo textový uzel. Postup vyhodnocení je následující. První se zpracuje identifikátor osy a určí uzly, které se budou v tomto kroku zpracovávat. Takto vybranou množinu uzlů omezíme pomocí testu uzlu, kterým může být třeba název elementu. V posledním kroku se zohlední dodatečné podmínky. Uzly, které vyhovují, postupují do dalšího kroku. Ten se pak bude vyhodnocovat postupně pro všechny uzly, které do něj postoupily (Fawcett, Quin, Ayers, 2012, s. 216).

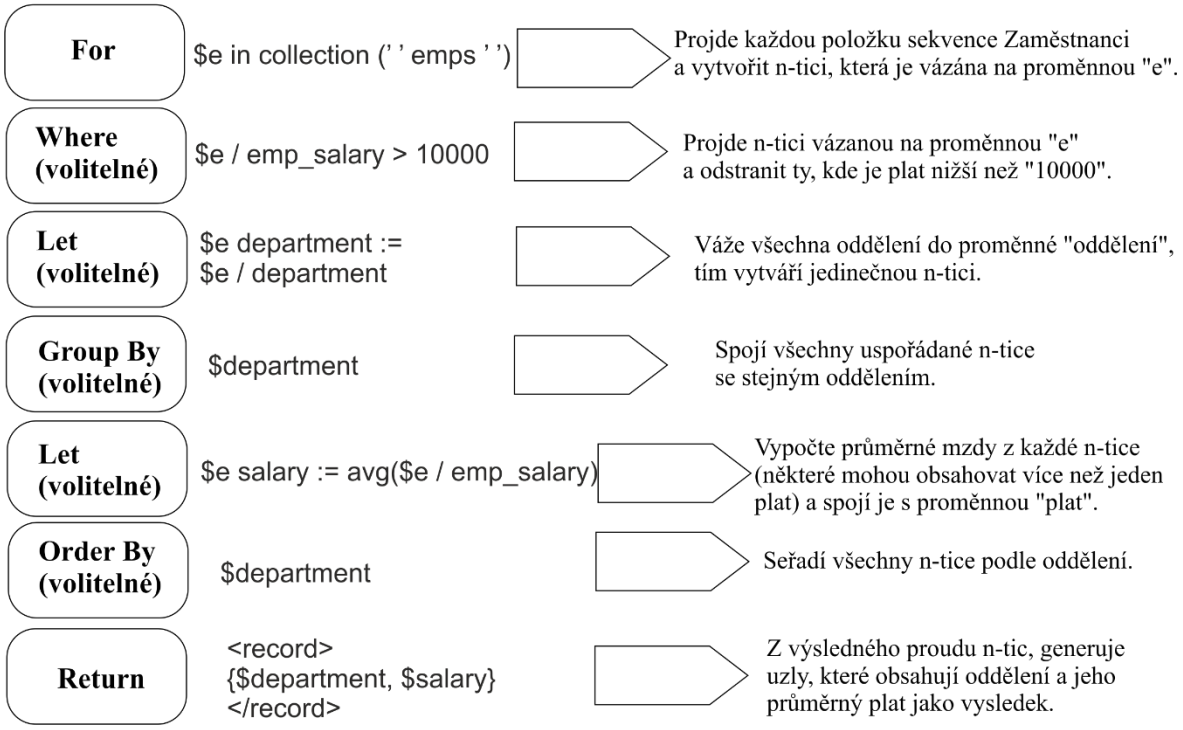
3.4.7 XQuery

XQuery je jazyk pro vyhledávání a manipulaci čehokoliv, co můžeme reprezentovat pomocí stromové struktury modelu. Původní záměr autorů (W3C) bylo vytvořit dotazovací jazyk nad daty ve formátu XML. Tím by se z XQuery jazyka stal ekvivalent jazyku SQL pro relační databáze, toto srovnání platí na úrovni XML databáze, ale jazyk XQuery našel uplatnění i v ostatních implementacích. Tohoto je dosaženo tím, že XQuery i Xpath (od verze 2) jsou postaveny kolem stejného abstraktního datového modelu XDM. Kvůli tomuto XQuery není definováno, aby operovalo přímo nad XML dokumenty, ale pracuje s abstraktem nazývaným datový model instancí. Tento model je možné získat z XML dokumentů, relačních databází, RDF schémat a geografických informačních systémů.

XQuery je silný datový programovací jazyk, jehož funkčnost přesahuje dotazování a manipulaci s XML daty. Jazyk poskytuje mechanismy i pro konstrukci XML dokumentů, mezi další funkce patří definice proměnných, rekurze a funkce.

Syntaxe jazyka XQuery je nesourodá a eklektická. Kombinuje v sobě tři různé styly, které adoptovala z odlišných zdrojů. První jsou výrazy z jazyka XPath 2.0. Každý validní výraz z jazyku XPath 2.0 je zároveň platným dotazem v XQuery. Druhým prostředkem jazyka XQuery jsou výrazy FLWOR viditelně inspirované jazykem SQL. Zkratka FLWOR shrnuje hlavní klauzule této části jazyka: For, Let, Where, Order by a Return. Na následujícím příkladu je jasně patrná ekvivalence s SQL dotazy.

```
for       $e in collection (' ' emps ' ')
where     $e / emp_salary > 10000
let       $e department := $e / department
group by  $department
let       $e salary := avg($e / emp_salary)
order by  $department
return    <record> {$department, $salary} </record>
```



Obrázek 3-4 FLOWR výraz

Poslední část syntaxe tvoří konstruktory XML. Ty mohou být literální, nebo počítané. Syntaxe dovoluje uživateli mnohonásobné vnořování výrazů FLWOR a konstruktorů XML, což vede ke tvorbě velice složitých dotazů.

3.5 Nativní XML databáze

Nativní XML databáze (zkratka NXD) je označení pro databázový systém, který splňuje následující body:

- Musí umožnit ukládat a získávat dokumenty odpovídající logickému modelu XML dokumentu. Příkladem těchto logických modelů jsou datový model XPath, XML Infoset a modely vycházející z DOM nebo SAX.
- Nemusí implementovat konkrétní model fyzického uložení dat. Může být postaven např. na relační, hierarchické nebo objektově orientované databázi.
- XML dokument by měl tvořit základní (logickou) jednotku databáze.

Produkt, který splňuje podmínky výše uvedené, je možné považovat za nativní XML databázi (Fawcett, Quin, Ayers, 2012, s 342). Kromě databází, které splňují podmínky výše, se můžeme setkat i s XML podporujícími databázemi. Tyto databáze umožňují ukládání XML dokumentů. Tuto podporu v určité formě nabízí většina relačních, objektových i hierarchických databází. U relačního modelu databáze se používají různé techniky pro uložení XML dokumentu do tabulek. Dokument se rozloží podle XML schématu a jednotlivé části jsou ukládány do tabulek nebo pro jeho uležení je použita speciální kolekce v systému zprávy databáze – CLOB (Character Large Object).

Roli standardního dotazovacího jazyku v XML databázích plní XQuery. Na rozdíl od SQL jazyka se nezabývá aktualizacemi dat, a proto vzniklo několik aktualizací. Nejvýznamnější a nejčastěji používané jsou XUpdate, vyvinutý organizací XML:DB a XQuery Update, který vyvinuli ve W3C. S těmito aktualizacemi je možné přidávat, mazat a upravovat libovolné části XML stromu (Powell, 2007, s. 285).

Indexovací metody používané pro XML dokumenty jsou následující:

- Strukturální index: Indexy elementů a atributů a umístění vzhledem k ostatním prvkům v rámci jednoho XML dokumentu. Umožňují vyhledávání podle jména a pozice elementu nebo atributu.
- Hodnotový index: Text a hodnoty atributů jsou často vyhledávané v rámci jednoho dokumentu XML. Jejich indexací se vyhledávání značně usnadní.
- Full-text index: Umožňuje vyhledávání specifických textových řetězců napříč jedním nebo mnoha dokumenty.

3.5.1 Přehled implementací

Krátký přehled nejznámějších implementací. Hlavní důraz se klade hlavně na open source řešení, ale zmíněny jsou i komerční produkty.

Databáze s podporou XML

- Open source licence
  - MySQL
  - PostgreSQL
  - OpenLink Virtuoso
- Komerční
  - Oracle
  - IBM DB 2
  - MS SQL server

Nativní XML databáze

- Open source licence
  - eXist
  - Berkeley DB XML
  - Sedna XML DBMS
  - BaseX
- Komerční
  - MarkLogic Server
  - X-Hive/DB

3.6 eXist

Pro testování jsem si vybral produkt eXist od Wolfganga Meiera. Tento projekt začal na přelomu 21. století na Technické univerzitě v Darmstadtu. Potřeba Wolfganga Meiera po systému, který bude schopen dělat analýzu a tvořit dotazy nad XML daty vedla k tvorbě vlastního díla, které pojmenoval eXist. Na začátku roku 2001 dokončil první verzi psanou v jazyce Java. Verze vycházela ze základů relačního databázového modelu, měla velice základní funkcionalitu a omezené indexování. V roce 2004 došlo k odstranění relačního modelu a nahrazení jej nativní XML podporou. Ve stejné roce došlo k využití eXist v prvním komerčním projektu. Během roku 2005 podpora všech uživatelů eXist umožnila Woflgangu Meierovi skončit na univerzitě a věnovat se jen dalšímu vývoji, přidalo se k němu několik dalších programátorů a v roce 2006 vydali verzi 1.0. Produkt se rapidně vyvíjel v následujících letech, byla zlepšena stabilita a transakční management. Z původního nástroje pro vyhledávání a dotazování na XML daty se pomalu stávala plnohodnotná XML databáze. Verze 1.4 byla vypuštěna v roce 2009, počet organizací používajících eXist stoupal. Další vývojové úsilí se zaměřilo na stabilizaci, opravování chyb a zvýšení spolehlivosti. Objevují se i první problémy se zpětnou kompatibilitou. Velké problémy působily nové aktualizace v jazyce XQuery, které ničily zpětnou kompatibilitu s již existujícími aplikacemi. Zatím konečná verze 2.0, z února 2013 přinesla masivní skok kupředu od 1,4. V červnu 2013 byla vydána verze 2.1, která hlavně adresovala chyby z verze

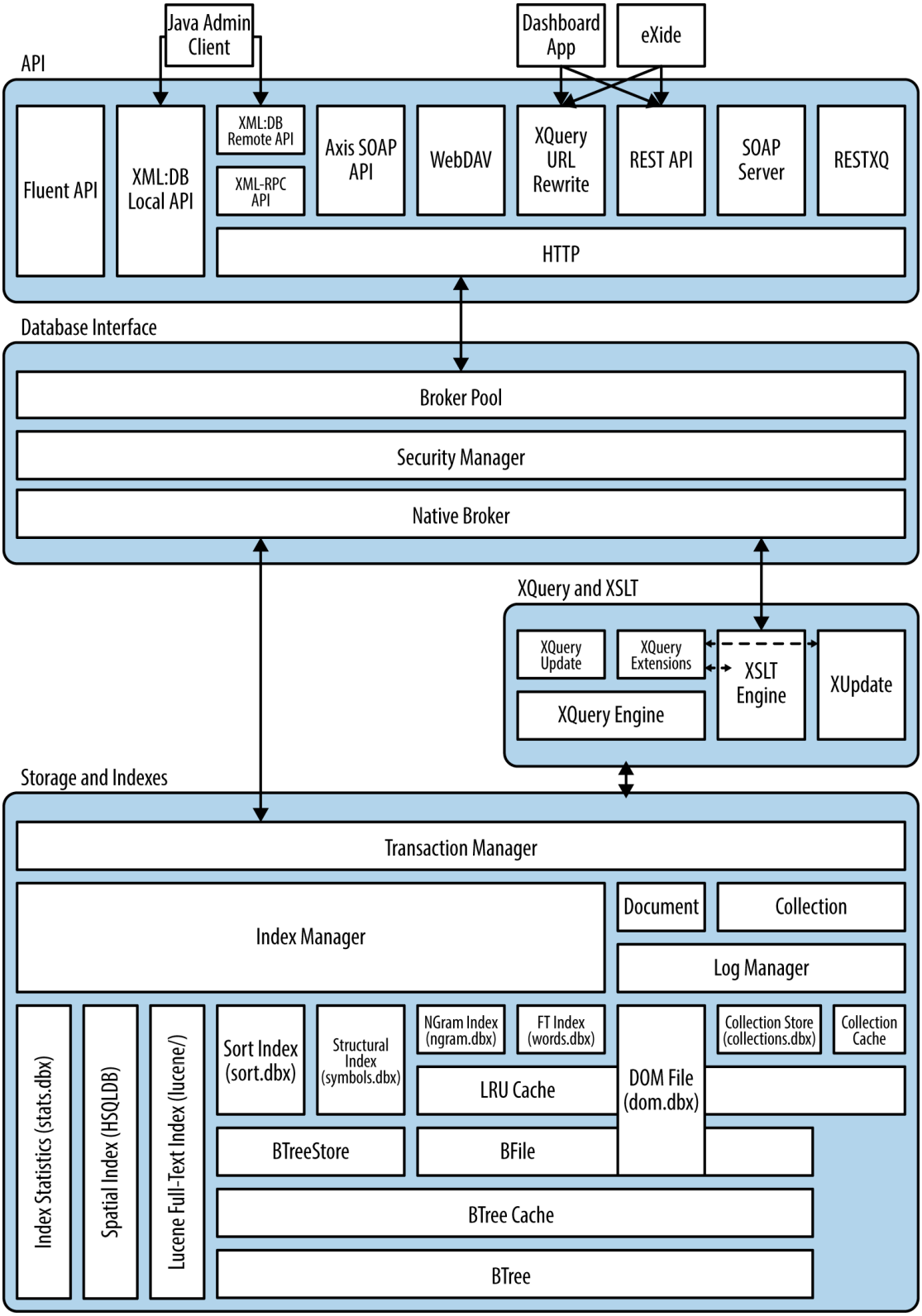
2.0. V únoru 2014 byla vydána zatím poslední verze 2.2. Budoucí vývoj eXist je směřován k zlepšení jádra produktu s možností separátních modulu pro rozšířenou funkcionalitu.

3.6.1 Licencování eXist

eXist je open source software napsaný v jazyce Java. Je volně šiřitelný, jak ve zdrojovém kódu, tak i v binární formě. eXist je k dispozici v rámci Lesser GNU Public License (LGPL). Počáteční vývoj byl hrazen sám autorem, později dobrovolnými příspěvky od uživatelů nyní je možnost zakoupit i placenou podporu pro určitý časový horizont. Kolem eXist software se vytvořila silná komunita lidí, jejíž cílem je usnadnit práci s eXist nováčkům a pomoci vývojářům (Siegel, Retter, 2014, s. 2).

3.6.2 Architektura eXist

Zde bude blokově nastíněna architektura softwaru eXist. eXist je za posledních 14 let převážně vyvíjen v programovacím jazyce Java. Rozšíření a přídavky (add-ons) jsou často psány v XQuery nebo XSLT. Bez ohledu na to, jak se rozhodneme nainstalovat a používat eXist, jeho architektura (viz obrázek 3-5) zůstává z velké části stejná, s různě volitelnými komponenty v závislosti na používání. Každé spojení z API do eXist využívá jedno vlákno a spolupracuje s *broker pool*, který je nakonfigurován s řadou *brokerů* (20 ve výchozím nastavení). Každý *broker* je vlákno, které spolupracuje s databází a představuje požadavek databáze. Požadavek je buď aktualizace databáze (add/delete/store/update), nebo dotazovací operace (XPath, XQuery nebo XSLT). Pokud dojde, k připojení s všechny *broker* vlákna jsou obsazené, dojde k pozastavení, dokud je nějaký neuvolní.



Obrázek 3-5 kompletní diagram architektury eXist (Siegel, Retter, 2014, s. 66)

eXist nabízí, tři možností pro finální nasazení. Tyto možnosti ovlivňují, jaké modely z výše uvedené architektury budou použity a jak budou vzájemně spolu komunikovat.

- Vložená (embedd) nativní XML databázová knihovna
- Nativní XML databázový server
- Platforma pro webovou aplikaci

Při vložení přímo do vlastní aplikace, eXist se chová jako standardní knihovna třetí strany. Uživatel v aplikaci procuje s funkcemi, které pracují nad třídami, pomocí nichž eXist provádí dané operace. Při přímém vložení eXist komunikuje pomocí dvou API, XML:DB Local API a Fluent API. Tato možnost nasazení je vhodná, pokud potřebujeme vytvořit samostatnou (standalone) aplikaci využívající pokročilé zpracovávání XML dokumentů a tvorbu dotazů nad nimi.

V klient/server architektuře je eXist nasazen jako centrální server a uživatelům je dán přístup do databáze pomocí WebDAV a XML:DB Remote rozhraních. WebDAV umožňuje uživatelům přímo manipulovat s dokumenty v databázi pomocí správce souborů v jejich operačním systému. XML:DB Remote dovoluje připojení do eXist pomocí Java Admin Client rozhraní, ze kterého je možné spravovat databázi a provádět dotazy.

Při vývoji webové aplikace obvykle vznikají dvě domény, ve kterých dochází ke zpracování kódu. Jde o stranu serveru a stranu klienta. Kód na straně serveru je prováděn na webové aplikaci samotného serveru (eXist), zatímco na straně klienta se kód provádí obvykle ve webovém prohlížeči. eXist může sloužit jako kompletní platforma pro webové aplikace, pokud jsou aplikace vyvinuty v jedné nebo více z následujících jazyků: XQuery, XSLT, XProc, a XForms. Po obdržení žádosti od klienta eXist zpracuje kód na straně serveru a generuje odpověď obsahující výsledky zpracování pro klienta. Reakce zaslána klientovi je nejčastěji v jedné nebo více z následujících formátů: HTML5, XHTML, XML, JSON, XForms, CSS, nebo různé binární formáty (např. obrázky pro zobrazení ve stránce). Při nasazení jako platformy pro webové aplikace existuje mnoho komponent, které mohou, ale nemusí být využívány v závislosti na webové aplikaci. Je doporučeno, zakázat ty, které nebudou používány. Bez ohledu na aplikaci, pokud potřebujeme používat eXist jako platformu pro webové aplikace, bude to vyžadovat použití Java web aplikačního serveru (Siegel, Retter, 2014, s. 68-70).

- eXist je vybaven Jetty Java web aplikačním serverem. Jetty může být použit s velmi malým úsilím a jedná se o doporučený postup, je dobře podporován ze strany komunity a vývojářů.
- Je možné postavit a nasadit eXist jako WAR soubor do Java web aplikačního serveru dle vlastního výběru (např. Apache Tomcat, GlassFish, JBoss Application Server, atd.). Toto je řešení pro organizace, které již investovali do konkrétní technologie. Je však vhodné provozovat eXist ve svém vlastním web aplikačním serveru, v opačném případě se bude dělit o zdroje s dalšími aplikacemi na stejném serveru.

Nyní bude blíže představena skladovací architektura. Pokud vložíme do eXist XML dokument, nejprve dojde k parsování dokumentu (dochází i k validaci, pokud je požadováno). Následně dojde k extrahování všech informací z dokumentu a jeho uložení. Zároveň proběhne indexování klíčových vlastnosti v dokumentu. Obecné schéma průběhu ukládání XML dokumentu je na obrázku 3-6. eXist neukládá dokumenty jako řadu XML souborů na disk, to není efektivní skladovací formát pro databázové operace. Místo toho je dokument rozložen na informace, které jej tvoří a následně je uložen do samostatných částí optimalizovaných binárních souborů na disku. Tento proces je transparentní pro uživatele, vždy je možné vyžádat původní XML dokument, dojde k jeho rekonstrukci a zaslání uživateli. Indexy, které vzniknou během ukládání XML dokumentu, umožňují později provádět rychlé a efektivní dotazy nad dokumenty v databázi.

3.6.5 Indexování v eXist

Spravovat obsáhlou databázi bez využití indexů je prakticky nemožné. Výkonnostní degradace při růstu datových souborů roste. Z tohoto důvodu nastavení správných indexů stojí za námahu a často bývá nezbytné. Systém řízení báze dat eXist obsahuje velké množství indexů. Dokonce je zde i možnost z využitím jazyku Java indexy měnit nebo definovat vlastní.

Strukturální indexy

Strukturální index sleduje stromové struktury uzlů ve všech dokumentech XML, které jsou uložené v databázi. Indexují se všechny prvky a atributy tak, že když je přijat dotaz jako „// title“, je možné rychle najít všechny vhodné tituly prvků. Indexace se provádí pomocí kvalifikovaného názvu uzlu. Strukturální index také automaticky indexuje

identifikující atributy. Jedná se o atributy xml: id nebo atributy výslovně označeny jako typ id v připojeném DTD. Strukturální index se používá pro řešení téměř všech XPath a XQuery výrazů. Nelze konfigurovat nebo vypnout, je nedílnou součástí databázového systému eXist.

Rozsahové indexy

Strukturální indexy sice pomáhají najít rychle určité uzly v XML dokumentu, ale nedělají nic se samotnými hodnotami uzlů (kromě xml: id). Zde přicházejí ke slovu rozsahové indexy, které jsou schopné pracovat nad hodnotami uzlů. Jsou využívány při použití =, >, <, porovnání v XPath výrazech a při porovnávání řetězců uvnitř funkcí. Index lze využít i na nezřetězené porovnávání. Je možné jej definovat nad hodnotami typu xs: double, xs: integer a xs: datetime.

NGram indexy

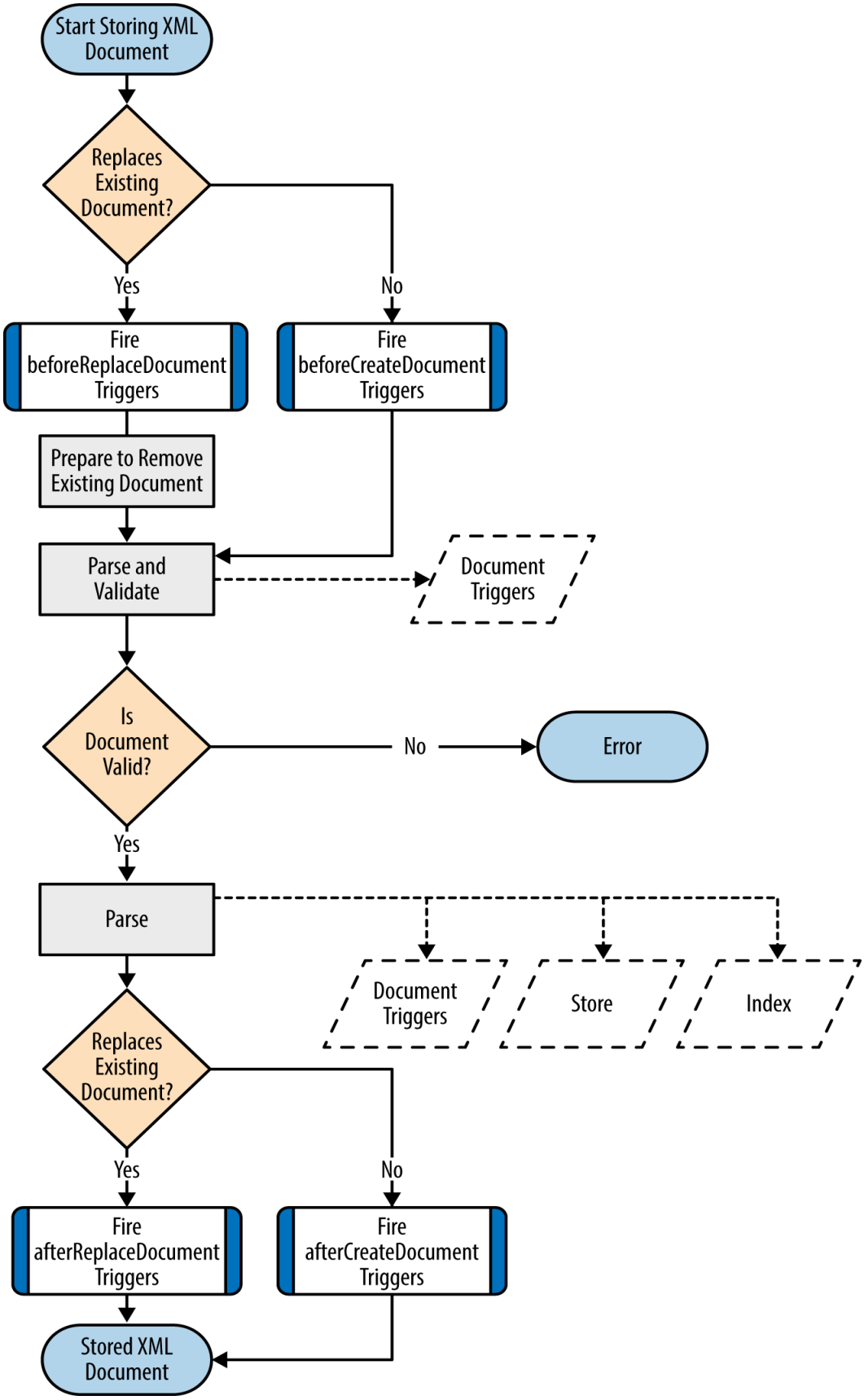
NGram indexy se používají k přesnému vyhledávání podřetězců. Například, pokud potřebujeme nalézt řetězec „def“ v „abcdefghij“ index NGram nám v tom může pomoci. Indexy zachovají mezery a interpunkci. Chceme-li použít tento index, je třeba funkce z rozšiřujícího ngram modulu. Indexované hodnoty jsou rozděleny do tokenů o n znacích, z tohoto plyne označení NGram. V databázi eXist je n v základu nastaveno na hodnotu tři, přináší to nejlepší kompromis mezi výkonem a velikostí indexu. Je možné tuto hodnotu změnit editací hodnoty v XML souboru v domovském adresáři $EXIST_HOME/conf.xml.

Fulltextové indexy

Fulltextové indexování umožňuje vyhledávat v dokumentech slova a fráze pomocí dotazovacího jazyka s funkcemi jako fuzzy shody a booleovské výrazy. Když se používají v kombinaci s "Klíčovými slovy kontextu" (KWIC), výsledky mohou být zobrazeny velmi rychle a ve snadno čitelné podobě.

3.6.4 Shrnutí

Ze všeho co bylo zmíněno výše je jasné, že produkt od Wolfganga Meiera a jeho spolupracovníků je multifunkční a zvládá daleko více než jen ukládání XML dokumentů. eXist je možné využít jako webový server, document search engine, platforma pro webovou aplikaci, platforma pro tvorbu a uchování dokumentů, vložený set knihoven pro vlastní užití ve vyvíjené aplikaci. Toto všechno se dá získat zdarma v rámci LGPL licence.



Obrázek 3-6 ukládání XML dokumentu (Siegel, Retter, 2014, s. 73)

4. Analytická část

V následující části bude vysvětlena základní problematika testování systémů řízení báze dat. Teoreticky budou nastíněny důvody, strategie, možná řešení a nástroje pro tvorbu srovnávacích testů. Obecně problematika výkonnostního testování systému řízení báze dat je velice specifická a vyžaduje velkou znalost testovaného systému, vše zde zmíněné mechaniky jsou spíše chápány jako základní postupy. Nakonec budou provedeny praktické testy na systémech řízení báze dat MySQL ve verzi 5.6 a eXist ve verzi 2.2. Nejprve je vysvětlena tvorba testovací datové základny a následně popsány testy, které se prováděli v obou systémech.

4.1 Výkonnostní testy

Výkonnostní testování (benchmarking) je výhodná a efektivní metoda jak zjistit, co se stane, když dáte systémů práci. Test může pomoci sledovat chování systému při zatížení, určit jeho kapacitu, zjistit důležité změny nebo objasnit, jak daná aplikace pracuje s různými daty. Testování umožňuje vytvářet fiktivní okolnosti, mimo hranice reálných podmínek. Zde je výčet základních benefitů testování:

- Ověřit předpoklady o systému a zjistit, zda jsou předpoklady realistické.
- Reprodukce špatného chování, které se snažíme v systému eliminovat.
- Měřit jak v současné době aplikace funguje. Pokud neznáme aktuální rychlost práce sledované aplikace, nemůžeme navrhnout vhodná zlepšení.
- Simulace větší zátěže, než na kterou byl systém projektován. Tím lze identifikovat úzké místo, na které narazíme, pokud se rozhodneme expandovat.
- Testování pomáhá při plánování budoucího rozvoje. Testy pomůžou odhadnout, jaký zvolit hardware, kapacitu sítě a další zdroje, které budou potřeba.
- Testovat způsobilost aplikace zvládat měnící se prostředí. Můžeme zjistit, jak se aplikace vyrovnává sice se sporadickými, ale velmi silnými nárazovými špičkami v souběžném zpracování, nebo s různými konfiguracemi serverů. Je možné testovat i reakce na odlišné rozložení dat.
- Vyzkoušet různý hardware, software a konfigurace operačního systému.
- Prokázat, že nově zakoupený hardware je správně nakonfigurován. Způsob pro odkrytí špatných komponent z výroby.

Problém při výkonnostních testech je správné určení pracovní zátěže. Zátěž, kterou během testování používáme je v porovnání se skutečnou zátěží značně zjednodušená. Skutečné pracovní podmínky jsou nedeterministické, proměnné a značně komplexní pro celkové porozumění. V důsledku toho se obvykle spokojíme s aproximací výsledných hodnot. Cílem je většinou zjistit, zda je tu stále nějaké množství volné kapacity v systému, které poskytne prostor pro budoucí rozvoj. Je možné udělat reálné zátěžové testování (na rozdíl od výkonnostního testování), ale vyžaduje to hodně starostí při vytváření datových podkladů a designování adekvátní pracovní zátěže. Výkonnostní testy se snaží být jednodušší, přímo srovnatelné navzájem, levnější a snadněji proveditelné. Během testování je velmi důležité mít jasno v tom, co je potřeba udělat a jakým způsobem aby výsledek byl smysluplný (Schwartz, Zaitsev, Tkachenko, 2012, s. 36, 37).

4.1.1 Strategie výkonnostních testů

Výkonnostní testy je možné rozdělit do dvou základních skupin, někdy označovaných za strategie výkonnostního testování. Můžeme testovat aplikaci jako celek, toto bývá označováno jako full – stack strategie nebo je možné testovat jednu komponentu (v našem případě databázi), to je označováno jako single component strategie. Je několik důvodů proč testovat celou aplikaci a ne jen její komponentu:

- Dochází k testování celé aplikace včetně webového serveru, kódu aplikace, sítě a databáze. Hlavní důraz je kladen na funkčnost celku.
- Databáze nemusí být vždy tím úzkým hrdlem, které určuje výkon celé aplikace. Full – stack testy to mohou potvrdit nebo vyvrátit.
- Je těžké popsat celkové chování aplikace, aniž by byla testována jako celek.

Základní nevýhodou je samozřejmě obtížnost návrhu full – stack testů a jejich správné provedení. Pokud je návrh nebo provedení špatné, může dojit na jejich základě k chybným rozhodnutím a to vše kvůli tomu, že testy nereflektovali skutečnost.

Někdy je však třeba testovat jen jednu komponentu výsledné aplikace, zde přicházejí na řadu testy se single component strategií. Pozornost bude věnována pouze těm testům, které se zaměřují na testování databázové komponenty. Tyto testy nám mohou, poskytnout odpovědi na následující:

- Chceme adresovat konkrétní problém, který vidíme v aplikaci.

- Potřebujeme porovnat výkonnostní rozdíly mezi různými schématy nebo dotazy.
- Raději provedeme několik menších testů a zhodnotíme dané výsledky a změny, než provádět full – stack test, kde je delší doba čekání na výsledky.

Je vhodné, aby při testování databáze docházelo k dotazování proti skutečné datové sadě. Samotná data a jejich velikost by měla být realistická. Pokud je to možné, je doporučené použít skutečné data. Nastavení realistického testu je velmi komplikované a časově náročné, získání kopie reálné datové sady je také značně komplikované, v mnoha případech i nemožné, jelikož taková databáze při vývoji nové aplikace ani nemusí ještě existovat. V těchto případech nezbývá než simulovat aplikační data i zátěž využitím substitutů.

4.1.2 Zaměření výkonnostních testů

Je doporučené identifikovat cíle ještě před vlastním testováním nejlépe ještě předtím, než dojde k návrhu kritérií samotného testu. Cíle, kterých chceme dosáhnout, určují nástroje a techniky, které budeme používat k získání smysluplných výsledků. Častým a v mnoha případech doporučeným postupem pro identifikování cílů je jejich formulace do otázek. Například „Je tento procesor výkonnější než ten předchozí?“ nebo „Získáme pomocí nové metody indexování vyšší rychlost dotazů?“ Výsledné cíle je možné rozdělit při testování databázových systému do následujících kategorií:

Počet transakcí za jednotku času (Throughput) je jeden z nejznámějších klasických testů výkonnosti databázových aplikací. Existují i standardizované testy výkonnosti, jako je například test TPC – C (viz http://www.tpc.org), mnozí vývojáři databázových systémů na těchto testech intenzivně pracují, za účelem získání vysokého hodnocení. Tyto testy výkonnosti měří výkon zpracování transakcí online (OLTP – online transaction processing) a jsou nejvíce užitečné pro interaktivní víceuživatelské aplikace. Obvyklou měrnou jednotkou je počet transakcí za sekundu (Schwartz, Zaitsev, Tkachenko, 2012, s. 38).

Doba odezvy (Latency) znázorňuje dobu, po kterou byla požadována nějaká úloha. V závislosti na aplikaci, bude nutné měřit čas v mikrosekundách, milisekundách, sekundách nebo v minutách. Z těchto údajů lze odvodit souhrnnou dobu odezvy, jako je průměr, maximum, minimum a percentil. Maximální doba odezvy je zřídka užitečný údaj, protože čím déle test běží, tím delší je maximální doba odezvy. Z tohoto důvodu je běžné používat percentily doby odezvy. Například pokud je 95. percentil doby odezvy 5 milisekund, s 95% pravděpodobností víme, že úloha skončí do 5 milisekund. Je užitečné výsledky těchto testů

převést do grafické podoby, a to buď jako spojnicový graf nebo jako bodový diagram, kde můžeme vidět, jak jsou výsledky distribuovány. Tyto grafy ukazují, jak se výsledky testů vyvíjejí v časovém horizontu.

Souběžnost (Concurrency) je důležitý, ale často nepochopený údaj. Například pokud nás zajímá kolik uživatelů současně prohlížení webové stránky ve stejný čas, bývá toto obvykle měřeno počtem naráz otevřených sessions. Nicméně protokol HTTP je bezestavový a většina uživatelů si jen čte co má ve svých prohlížečích, takže z tohoto údaje nelze vyvodit žádnou souběžnou zátěž pro webový server. Stejně tak souběžnost na webovém serveru se nemusí nutně promítnout do databázového serveru. Jediná věc, se kterou přímo souvisí, je kolik dat mechanismus relačního úložiště musí být schopen zvládnout. Přesnějším měřením souběžnosti na webovém serveru je kolik souběžných požadavků na serveru běží v daném okamžiku. Při testování na souběžnost je důležité se pokusit určit pracovní souběžnost nebo počet vláken, spojení, které dělají práci současně. Zjistit zda se propustnost sníží nebo doba odezvy zvýší, když souběžnost vzroste. Pokud tomu tak je, aplikace pravděpodobně nezvládá špičky v zatížení. Testování souběžnosti je velmi odlišné od výše zmíněných ukazatelů. Obvykle po vyhodnocení nezískáme výsledek, ale spíše vlastnost toho, jak jsme test připravili. Místo měření souběžnosti, které se na aplikaci může dosáhnout, tak obvykle dochází ke generování různých úrovní souběžnosti pro potřeby testu, při kterých se měří výkon aplikace.

Škálovatelnost (Scalability) je vhodné určit pro systémy, které musí udržet určitou úroveň výkonu i pod měnící se pracovní zátěží. Výkon se typicky měří pomocí metriky, jako je propustnost nebo doba odezvy, přičemž pracovní zátěž může kolísat na základě změn velikosti databáze, počtu simultánních připojení, nebo hardwaru. Testy škálovatelnosti jsou dobré při plánování kapacit, protože mohou poukázat na takové slabiny v dané aplikaci, které by jiné testy nemuseli odhalit. Pokud například máme systém, který si vede dobře v testech výkonnosti a doby odezvy při jediném připojení, aplikace může mít špatný výkon v okamžiku, kdy dojde k jakémukoli stupni souběžného zpracování. Tento nedostatek v návrhu odhalí pouze takový test, který bude hledat konzistentní doby odezvy při zvyšujícím se počtu připojení (Schwartz, Zaitsev, Tkachenko, 2012, s. 39, 40).

V konečném důsledku je nejlepší testovat vše co je důležité pro vaše uživatele. Je dobré pokusit se shromáždit některé požadavky (formální i neformální) o tom, jaké jsou přijatelné

časy odezvy, jak velkou souběžnost můžeme očekávat a tak dále. S těmito daty je možné navrhnout svá kritéria tak, aby splňovaly požadavky. Je velmi důležité se v této fázi vyhnout tzv. tunelovému vidění, kdy dochází k zaměření na jednu komponentu a vypustí se požadavky na vše ostatní.

4.2 Postupy výkonnostních testů

Základní metodika tvorby výkonnostních testů byla zmíněna v předcházejících kapitolách. Dále budou nastíněny specifické postupy a obvyklá řešení, tyto postupy vycházejí z testování relačních databázových strojů. Testování nativních XML databází není nijak standardizovaný postup z důvodů obecného stáří technologie, většina metodik, které hodlám dále popsat, je možné využit i pro XML databáze. Předtím než se zmíníme jak navrhovat výkonnostní testy správně, je dobré si projít základní chyby:

- Použití neadekvátní velikosti dat, například provádět testy s jedním gigabytem dat, zatímco aplikace bude muset zvládat stovky gigabytů dat.
- Testuje se výkon distribuované aplikace na jediném serveru.
- U víceuživatelské aplikace se použije jednouživatelský testovací scénář.
- V cyklu se pouštějí identické dotazy. Dotazy ve skutečném světě nejsou úplně identické, což znamení, že cache dotazů nebývá využita úplně naplno. Identické dotazy jsou obvykle plně, nebo alespoň částečně vraceny z cache.
- Ignoruje se výkonnost systému ve chvíli, když ještě není dostatečně „zahřátý“. Netestovat po restartu serveru, nechat systém plně naběhnout.
- Zapomíná se na kontrolu chyb. Po výkonnostních testech je vždy dobré zkontrolovat chybové logy (Schwartz, Zaitsev, Tkachenko, 2012, s. 40, 41).

Vyvarování se těmto chybám během testů je základ k cestě za kvalitním výsledkem. Obvykle bychom se měli snažit, aby testy byly realistické, jak jen to je možné.

Prvním krokem při plánování výkonnostních testů je identifikovat problém a cíl. Dále rozhodnout, zda použít standardní test nebo navrhnout vlastní test. Při použití standardního testu je dobré se ujistit, že byl vybrán ten, který vyhovuje všem potřebám. Navržení vlastního testu je složitý a časově náročný proces. Na začátku je třeba pořídit vzorek datové sady, pokud to situace dovoluje. V dalším kroku spustíme proti pořízeným datům dotazy. Pro tento

krok je doporučeno nejdříve pořídit list dotazů, které se provádějí nad databází za určitý časový rámec (např. během hodiny, kde je nejvíce databáze vytížená, nebo za celý den). Tento postup umožní pokrýt systémové aktivity lépe, než opakování určité sady dotazů za pevně daný časový úsek. Při tvorbě vlastního testu je doporučené dokumentovat každý krok, pokud bude třeba spustit námi navržený test vícekrát, je třeba, aby se test dal přesně reprodukovat. Také to může sloužit jako plán pro ostatní lidi, kteří budou mít zájem daná test provádět. Plán by obecně měl obsahovat způsob získání testovaných dat, kroky potřebné k nastavení systému pro testování, způsob jakým byly změřený výsledky testu a jestli byly dále analyzovány.

4.2.1 Délka výkonnostních testů

Výkonnostní testy se obvykle provádějí po určitou smysluplnou dobu. Při testování se snažíme dostat systém do ustáleného stavu výkonnosti a až poté začít testovat. Dosažení ustáleného stavu může trvat překvapivě dlouhou dobu a především na serverech s množstvím dat a velkou paměti. Většina systémů je vybavena vyrovnávací pamětí, která pomáhá absorbovat výkonnostní špičky. Pokud chceme otestovat plný potenciál systému, je třeba navrhnout test tak aby kompenzoval schopnosti vyrovnávacích pamětí a ukázal pravé schopnosti. Velmi častou chybou je použití série krátkých testů, například 60 sekundový test a vyvodit závěry o výkonnosti systému z tohoto měření. Pokud chceme zachytit skutečný výkon daného systému pomocí výkonnostních testů, je třeba je nechat provádět tak dlouho, dokud nedosáhneme ustáleného výsledku, poté je dobré tyto testy pravidelně opakovat a porovnat ustálené výsledky mezi sebou. Během testu se snažíme zachytit co nejvíce informací o systému. Vytvořit si srovnávací adresáře pro naměřená data s podadresáři pro výsledky každého průběhu testu je doporučený postup. Můžeme umístit výsledky, konfigurační soubory, měření, skripty a poznámky pro každý průběh do příslušného podadresáře. Pokud je možné naměřit více informací, než z daného testu potřebujeme získat, je dobré si je ponechat pro možné budoucí upotřebení. Je mnohem lepší mít nepotřebná data, než postrádat data potřebná.

4.2.2 Průběh testů

Obvykle se proces testu automatizuje. Zlepšuje to výsledky a zamezuje to pozměňování kroků při opakování testu. Skriptovací jazyky jako jsou: shell, PHP, Perl dokážou s automatizací testu pomoct. Automatizuje se co nejvíce procesů včetně načítání dat, běh samotného testu a záznam výsledků. Test se obvykle spouští několikrát. Počet

pokusů záleží na metodologii hodnocení a důležitosti výsledků. Pokud potřebujeme dosáhnout určité jistoty, je třeba spustit test vícekrát. Obvyklým postupem je najít nejlepší výsledek, průměr ze všech výsledků, nebo se průměrují nejlepší hodnoty. Jestliže test uspokojivě odpovídá na kladené otázky, proveďte jej několikrát po sobě a porovnejte získané výsledky. Pokud se budou lišit, opakujte test ještě několikrát nebo jej nechte vykonávat po delší dobu, tímto postupem obvykle dojde ke snížení rozptylu. Jakmile získáme potřebné data, nastane čas na jejich analýzu. Tímto procesem přeměníme data na informace. Hlavním cílem je získat odpověď na otázku, kterou jsme zastřešili celý test. Zda bude možné potvrdit hypotézu: „Pokud dojde k upgradu na čtyři výkonnější procesory, zvýší se propustnost o 50%, přičemž latence zůstane stejná“.

4.3 Nástroje pro výkonnostní testy

Návrh a tvorba vlastních výkonnostních testů vyžaduje mnoho času a důkladnou znalost potřebné problematiky. Pokud nemáme výše zmíněné, nebo jen opravdu není důvod pro tvorbu vlastní metodiky, je možné využít některý z dostupných nástrojů pro výkonnostní testy. V současnosti existuje široká paleta nástrojů, které jsou připraveny k okamžitému použití. V následují části práce, budou stručně popsány některé nástroje.

4.3.1 Full – stack nástroje

Stejně jako existují strategie výkonnostních testů, tak existují i nástroje, které se těmito strategiemi řídí. Testování za pomoci full – stack nástroje je obvykle nejlepší způsob jak získat celkový přehled o výkonu aplikace. Často požívané nástroje jsou tyto:

ab

ab je Apache HTTP server nástroj pro výkonnostní testy. Zobrazuje kolik požadavků za sekundu HTTP server je schopný obsloužit. Pokud se testuje webovou aplikaci, tento nástroj může zjistit, kolik požadavků za sekundu aplikace zvládne uspokojit. Jde o velmi jednoduchý nástroj s omezenou užitečností. Více informací o ab je k dispozici na adrese http://httpd.apache.org/docs/2.0/programs/ab.html.

http_load

Tento nástroj je podobný ab. Umožňuje ale některé funkce navíc. Je možné vytvořit vstupní soubor s různými URL, nástroj pak bude náhodně vybírat z tohoto souboru. Nástroj také umožňuje nastavení času mezi jednotlivými požadavky. Více informací na http://www.acme.com/software/http_load/.

JMeter

JMeter je Java aplikace, která umožňuje nahrát jinou aplikaci a měřit její výkon. Byl navržen pro testování webových aplikací, ale lze také použít pro otestování FTP serveru a zasílání dotazů na databázi přes JDBC. JMeter je mnohem složitější, než ab a http_load, má grafické uživatelské rozhraní s vestavěným vykreslováním grafů. Nabízí možnost nahrávat a přehrávat záznamy v offline režimu. Více informací na http://jakarta.apache.org/jmeter/.

XMark

Skládá se z jednoduchého nástroje xmlgen pro generování syntetického XML dokumentu, DTD schéma a sady 20 XQuery dotazů nad tímto dokumentem. Aplikace umožňuje jen změnu velikosti vstupních souborů. Více na http://www.xml-benchmark.org/.

TPoX

Komplexní nástroj pro testování nejen dotazovacího jazyka XQuery, ale i ukládání dat, indexování, podporu pro XML schémata, aktualizace pomocí XUpdate a transakce. Dokumenty, na kterých se testování provádí, jsou datově orientované a reprezentují finanční transakce. Kromě generátoru XML dat, schémat, dotazů a transakcí obsahuje také v Javě napsanou aplikaci, která umožňuje simulovat několik paralelně pracujících uživatelů. Vyvinuto společností IBM, http://tpox.sourceforge.net/.

4.3.2 Single component nástroje

Zde je několik užitečných nástrojů pro testování výkonu MySQL a systému, na kterém běží.

mysqlslap

Nástroj simuluje zatížení na serveru a reportuje informace zpět uživateli. Je součástí MySQL 5.1 server distribuce. Umožňuje nastavit počet souběžných připojení a buď provést SQL dotaz z příkazové řádky, nebo je možné vložit soubor obsahující SQL příkazy, které spustí. Více na http://dev.mysql.com/doc/refman/5.1/en/mysql-benchmarks.html.

MySQL Benchmark Suite

MySQL je distribuován s vlastní sadou pro výkonnostní testování. Pomocí této sady je možné měřit výkon několika různých databázových serverů. Měří, jak rychle server provádí dotazy. Podle výsledků je možné zjistit, které typy operací serveru dělají problémy. Hlavním přínosem této sady je, že obsahuje velké množství předdefinovaných testů, které jednoduše

aplikují a umožňují snadno porovnat různé úložné enginy nebo konfigurace serveru. Tímto nástrojem lze získat měřítko pro porovnání celkové výkonnosti serverů. Umožňuje spustit také podmnožinu testů (například jen otestovat výkonnost příkazu). Testy jsou většinou orientovány na procesor, ale obsahují i krátké časové úseky, které otestují rychlosti I/O na úložném médiu. Nevýhodou tohoto testu je omezená sada dat, která je pevně daná. Nelze tudíž testovat specifická data od uživatelů. Je nutné mít ovladače na Perl a DBD pro chod tohoto nástroje. Dokumentace je dostupná na http://dev.mysql.com/doc/en/mysql-benchmarks.html/.

Super Smack

Super Smack (https://github.com/tmountain/Super-Smack/) je nástroj pro výkonnostní testování, zátěžové testy a generování zátěže pro MySQL a PostgreSQL. Jedná se o komplexní, výkonný nástroj, který umožňuje simulovat více uživatelů, nahrání testovaných dat do databáze a zaplnění tabulek náhodně generovanými daty. Výkonnostní testy jsou obsaženy v "smack" souborech, které používají jednoduchý jazyk pro definování klientů, tabulek, dotazů, a tak dále.

sysbench

Sysbench (https://launchpad.net/sysbench) je vícevláknový systém pro výkonnostní testování. Zachycuje výkonu systému v termínech důležitých pro provoz databázového serveru. Umožňuje měřit výkon I/O, rychlost plánovače OS, přidělování paměti, POSIX vlákna a samotnou databázi na serveru. Sysbench podporuje skriptovací jazyk Lua (http://www.lua.org). Je to velmi oblíbený testovací a srovnávací nástroj pro MySQL, operační systém a výkonnost hardwaru.

ToXgene

Nástroj generuje XML dokument pomocí šablon v jazyce TSL (Template Specification Language). Tento jazyk vychází z XML Schema. Nástroj je využíván v sadách testů XBench a TPoX.

4.4 Testování MySQL a eXist systému řízení báze dat

Pro testování rychlosti dotazů byly zvoleny dva open source systémy řízení báze dat. Představitel relačního modelu je systém MySQL ve verzi 5.6.2, systém řízení báze dat je spravován pomocí softwarového nástroje phpMyAdmin ve verzi 4.2.11 a běží na webovém

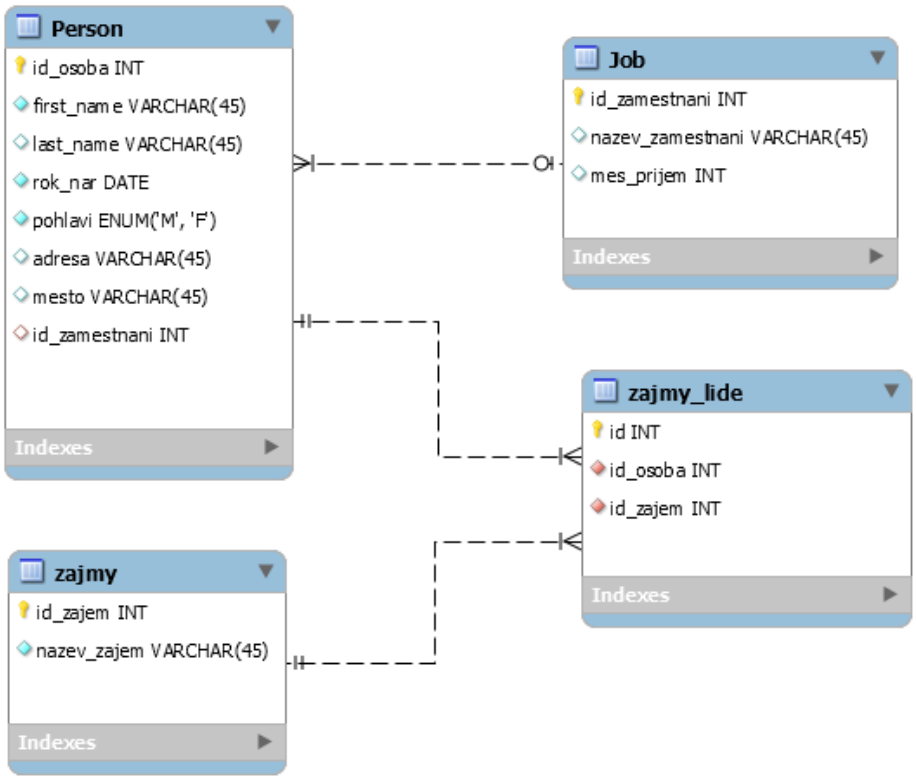
serveru Apache verze 2.4.10. Zástupce nativní XML databáze je výrobek eXist ve verzi 2.2, veškeré postupy spojené s prací v této databázi byly realizovány pomocí webového GUI dashboard a nástroje Java Admin Klient. Sestava, na které byly prováděny testy má následující specifikaci:

Operační systém	Windows 7 Home 64-bit, servise pack 1
Procesor	Intel® Core™ i5-2500K (6MB cache, 3.30 GHz)
RAM	Kingston HyperX Fury Black 8GB (2x4GB) DDR3 1866Mhz
HDD	Intel SSD 530 - 180GB

Tabulka 4-1 Hardwarová specifikace testovací sestavy

4.4.1 Datové podklady pro systém MySQL

Pro účely testování se nepodařilo opatřit reálné datové podklady, byl proto zvolen postup tvorby vlastního datového schématu, který byl následně naplněn náhodně generovanými daty. Základní návrh byl přetvořen do podoby EER (enhanced entity–relationship) diagramu pomocí vizuálního nástroje pro architekturu databází MySQL Workbench ve verzi 6.2.4, tento návrh je na obrázku 4-1.



Obrázek 4-1 EER diagram

Takto navržená datová struktura byla importována do databáze MySQL pomocí propojovacího nástroje. Jelikož testovací datový model počítá s využitím cizích klíčů, bylo rozhodnuto o využití úložného enginu InnoDB. V rámci testování byly nejdříve provedeny testy se základní indexací (indexování dle primárního klíče). Druhý testovací cyklus byl pak prováděn s indexy, které byly vytvořeny navíc. Pro generování dat byl použit program Spawner 0.2.3, pomocí něhož vzniklo 10 000 a 100 000 náhodně generovaných záznamů pro tabulku Person a Zajmy_lide. Tyto tabulky byly uloženy do dvou databázích s názvy emp pro 10 000 záznamů v tabulce Person a emp2 pro 100 000 záznamů v tabulce Person. V následujících tabulkách je možné vidět ukázky dat, které byly uloženy do databáze MySQL, u všech tabulek půjde jen o prvních 5 řádků.

id_zamestnani	nazev_zamestnani	mes_prijem
1	učitel	25000
2	pošťák	15000
3	soustružník	17000
4	hasič	20000
5	programátor	45000

Tabulka 4-2 vzorek dat z tabulky Job

id_zajem	nazev_zajem
1	3D modelování
2	atletika
3	divadlo
4	dřevořezba
5	fotografie

id	id_osoba	id_zajem
1	500	20
2	501	18
3	502	19
4	503	5
5	504	17

Tabulka 4-3 vzorek dat z tabulek Zajmy a Zajmy_lide

id_osoba	first_name	last_name	rok_nar	pohlavi	adresa	mesto	id_zamestnani
1	Allen	Mcgowan	1970-12-17	M	75995 West Iran, Islamic Republic of St.	Albuquerque	9
2	Keefe	Humphrey	1986-03-26	M	68994 Turkey Ct.	Morrison	1
3	Christopher	Foster	1987-09-24	M	1454 Bouvet Island Blvd.	Sherrill	10
4	Porter	Farley	2005-04-11	M	9391 South Hazleton Way	North Little Rock	8
5	Stuart	Woods	1995-02-22	M	45007 West Reunion Ct.	Hickory	10

Tabulka 4-4 vzorek dat z tabulky Person

4.4.2 Datové podklady pro systém eXist

Aby bylo možné alespoň přibližně srovnat výsledky měření prováděného na dvou odlišných systémech řízení báze dat, bylo třeba je provádět nad podobnou množinou uložených dat. Bylo třeba exportovat data uložená v relační databázi MySQL do formátu XML a ten uložit do nativní XML databáze eXist. Zvoleny byly následující dva postupy. U prvního postupu byla vyvinuta maximální snaha o zachování datového schématu definovaného pro relační model, tudíž byly vytvořeny čtyři samostatné XML soubory jako obraz čtyř tabulek v MySQL. Druhý postup vybere všechny data ze všech čtyřech tabulek (výchozí tabulka je Person) pomocí SQL příkazů JOIN a data se následně uloží do jednoho XML datového dokumentu. XML dokumenty budou tvořeny na základě XML schématu, které bude odvozeno dle datových typů a struktury relačního návrhu, tímto postupem vznikne datově orientovaný dokument.

Postup exportu dat z MySQL a tvorby XML dokumentu:

- Vytvoření XML schémat, v našem případě bylo vytvořeno pět schémat, pro každou tabulku po jednom schématu a jedno pro všechny data dohromady. XML schéma pro spojená data je na obrázku 4-2. Ostatní schémata vychází z tohoto, akorát byla převzata struktura cizích klíčů z relačního modelu.
- Pomocí rozhraní phpMyAdmin byly z každé tabulky zvlášť exportována data do formátu CSV pro MS Excel, ve kterém byla následně propojena s příslušným XML schématem a uložena jako XML dokument. Tímto postupem byly vytvořeny dokumenty: „Job.xml“, „Person.xml“, „Zajmy.xml“ a „Zajmy_lide.xml“. Tento postup byl aplikován jen s první databází v MySQL, která obsahuje 10 000 záznamů v tabulce Person.
- V rozhraní phpMyAdmin byl vykonán SQL příkaz pro výběr dat ze všech čtyřech tabulek, tento výběr byl exportován do souboru CSV pro MS Excel. V tabulkovém editoru Excel byl tento datový soubor upraven a po vložení XML schématu bylo možné data uložit jako XML dokument s definovanou strukturou dle XML schématu.

  Příkaz pro výběr dat měl tuto podobu:

  ```
  SELECT * FROM person
  LEFT JOIN job ON person.id_zamestnani = job.id_zamestnani
  LEFT JOIN zajmy_lide ON person.id_osoba = zajmy_lide.id_osoba
  LEFT JOIN zajmy ON zajmy_lide.id_zajem = zajmy.id_zajem;
  ```

Výstupem byly dokumenty: „emp10k.xml“ a „emp100k.xml“.

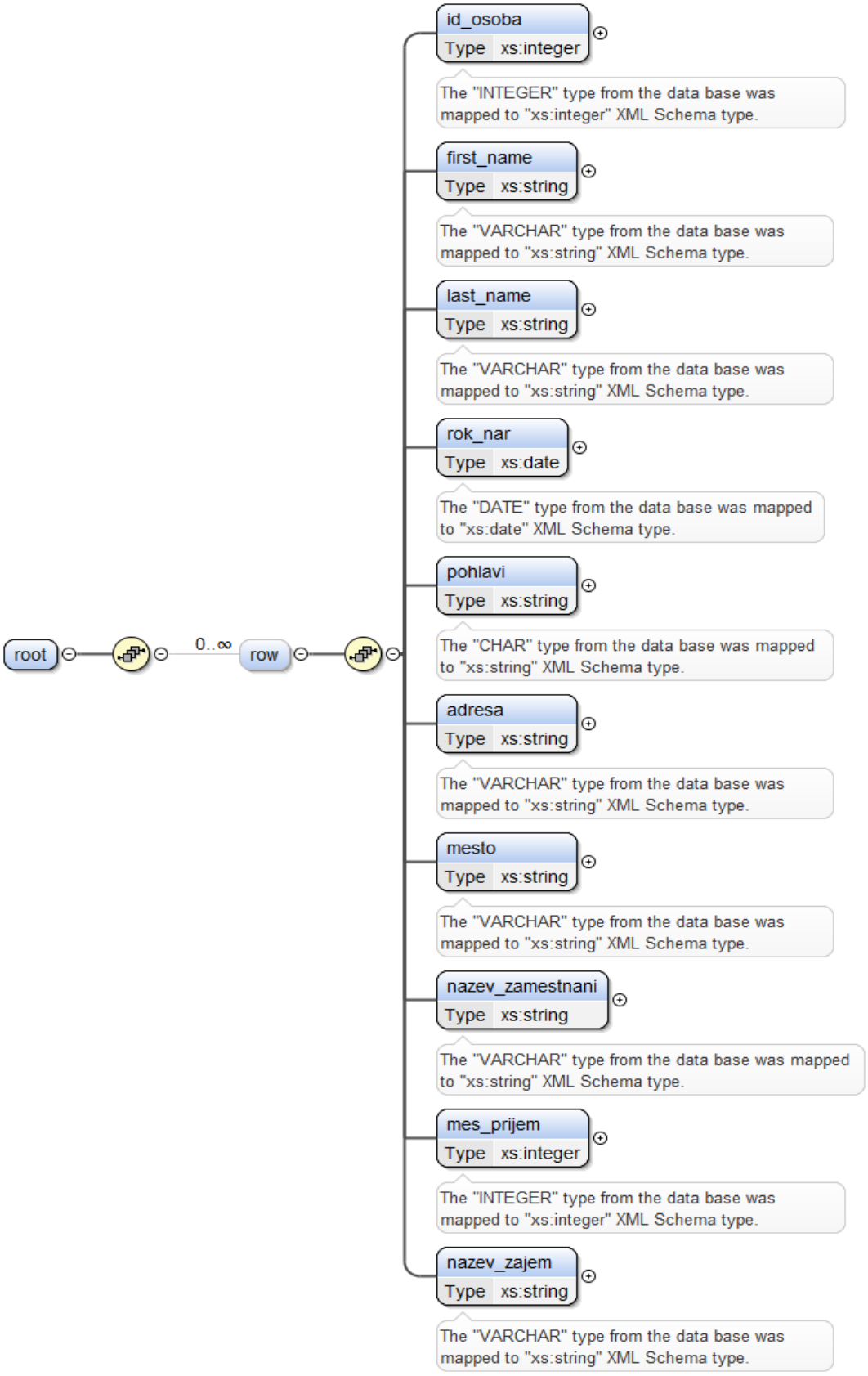
- Pomocí webového GUI dashboard pro eXists databázi byly vytvořeny kolekce, do kterých se následně nahrály dokumenty vytvořené v předchozím kroku. XML dokumenty měly tyto přístupové cesty „exist:/db/apps/test“ a „exist:/db/apps/test2“. Kolekce „test“ obsahuje data, která byla sloučena a poté exportována do XML dokumentu, kolekce „test2“ obsahuje data, jenž byla přímo exportována do XML bez předchozího sloučení.

4.4.3 Testovací dotazy pro systém MySQL

Datová struktura získaná v předchozích krocích bude testována za pomoci SQL dotazů. Dotazy byly navrženy tak, aby pokryly základní funkcionalitu jazyka SQL pro manipulaci s daty. Pro získání doby trvání jednotlivých dotazů je využito vestavěné profilovací funkce v systému MySQL. Profilování zadaného dotazu je třeba před dotazem aktivovat, toho je docíleno příkazem „SET profiling = 1;“ a po vykonání dotazu na určitá data je výsledek profilování zobrazen pomocí „SHOW PROFILES;“ příkazu. Pro eliminování vlivu cache paměti na vykonávané dotazy je použita klauzule „SQL_NO_CACHE“. Jednotlivé dotazy byly pojmenovány Test 1-12 pod stejným označením vystupují i v datových podkladech a výsledných grafech. První tři dotazy jsou definovány jen nad jednou tabulkou (Person) a kombinují funkce selekce a projekce z relační algebry. Zbylé dotazy pracují s více tabulkami a je v nich užito různých způsobů spojování.

Test 1 - dotaz vrací jména všech mužů a jejich věk srovnané dle abecedy podle příjmení
```
SET profiling = 1;
SELECT SQL_NO_CACHE CONCAT_WS(" ", last_name, first_name) AS cele_jmeno, TIMESTAMPDIFF(YEAR,rok_nar,CURDATE()) AS věk
FROM person
WHERE pohlavi = 'M'
ORDER BY cele_jmeno ASC;
SHOW PROFILES;
```

Test 2 - dotaz vrací jména všech osob narozených 1.1.2000 a později, srovnaných dle data narození
```
SET profiling = 1;
SELECT SQL_NO_CACHE CONCAT_WS(" ", first_name, last_name) AS cele_jmeno, rok_nar
FROM person
WHERE rok_nar >= '2000-01-01'
ORDER BY rok_nar ASC;
SHOW PROFILES;
```



Obrázek 4-2 XML schéma pro tvorbu spojeného XML dokumentu

Test 3 - dotaz vrací jména a věk všech lidí v pořadí dle jejich věku
```
SET profiling = 1;
SELECT SQL_NO_CACHE CONCAT_WS(" ", first_name, last_name) AS cele_jmeno, rok_nar, CURDATE(), TIMESTAMPDIFF(YEAR,rok_nar,CURDATE()) AS věk
FROM person
ORDER BY věk ASC;
SHOW PROFILES;
```

Test 4 - dotaz vrací jména všech lidí, co jsou zaměstnáni jako hasič
```
SET profiling = 1;
SELECT SQL_NO_CACHE person.first_name, person.last_name, nazev_zamestnani
FROM person JOIN job ON person.id_zamestnani = job.id_zamestnani
WHERE nazev_zamestnani = 'hasič';
SHOW PROFILES;
```

Test 5 - dotaz vrací jména a název zaměstnaní všech lidí v tabulce Person, srovnáno dle abecedy podle názvu zaměstnaní
```
SET profiling = 1;
SELECT SQL_NO_CACHE person.first_name, person.last_name, nazev_zamestnani AS zamestnani
FROM person JOIN job ON person.id_zamestnani = job.id_zamestnani
ORDER BY zamestnani;
SHOW PROFILES;
```

Test 6 - dotaz vrací jména, rok narození pro všechny kdo jsou zaměstnáni jako hasič a jsou narozeni 1.1.1980 a později
```
SET profiling = 1;
SELECT SQL_NO_CACHE person.first_name, person.last_name, person.rok_nar AS 'rok narození'
FROM person JOIN job ON person.id_zamestnani = job.id_zamestnani
WHERE nazev_zamestnani = 'hasič' AND person.rok_nar >= '1980-01-01'
SHOW PROFILES;
```

Test 7 - dotaz vrací jména všech zaměstnaných a název profese, seřazeno dle názvu zaměstnání
```
SET profiling = 1;
SELECT SQL_NO_CACHE nazev_zamestnani AS zamestnani, first_name, last_name
FROM person RIGHT JOIN job ON  person.id_zamestnani = job.id_zamestnani
ORDER BY zamestnani;
SHOW PROFILES;
```

Test 8 - dotaz vrací jména všech zaměstnaných i nezaměstnaných a název profese, seřazeno dle názvu zaměstnání
```
SET profiling = 1;
SELECT SQL_NO_CACHE nazev_zamestnani AS zamestnani, first_name, last_name
FROM person LEFT JOIN job ON  person.id_zamestnani = job.id_zamestnani
ORDER BY zamestnani;
SHOW PROFILES;
```

Test 9 - dotaz vrací počet lidí živících se jako hasič
```
SET profiling = 1;
SELECT SQL_NO_CACHE COUNT(*) AS 'Počet hasičů'
FROM person JOIN job ON person.id_zamestnani = job.id_zamestnani
WHERE nazev_zamestnani = 'hasič';
SHOW PROFILES;
```

Test 10 - dotaz vrací jména lidí a jejich zájmů
```
SET profiling = 1;
SELECT SQL_NO_CACHE person.first_name, person.last_name, zajmy.nazev_zajem, person.mesto
FROM zajmy_lide
JOIN zajmy ON zajmy_lide.id_zajem = zajmy.id_zajem
JOIN person ON zajmy_lide.id_osoba = person.id_osoba;
SHOW PROFILES;
```

Test 11 - dotaz vrací jména lidí, kteří se věnují rybaření
```
SET profiling = 1;
SELECT SQL_NO_CACHE  person.first_name, person.last_name, zajmy.nazev_zajem
FROM zajmy_lide
JOIN zajmy ON zajmy_lide.id_zajem = zajmy.id_zajem
JOIN person ON zajmy_lide.id_osoba = person.id_osoba
WHERE nazev_zajem='rybaření';
SHOW PROFILES;
```

Test 12 - dotaz vrací počet žen věnující se zájmu s 20 a více členy
```
SET profiling = 1;
SELECT SQL_NO_CACHE nazev_zajem, COUNT(*)
FROM person JOIN zajmy_lide USING (id_osoba) JOIN zajmy USING (id_zajem)
WHERE pohlavi = 'F'
GROUP BY nazev_zajem
HAVING COUNT(*) >= 20;
SHOW PROFILES;
```

Testy pro datovou strukturu se 100 000 záznamy v tabulce Person mají stejnou syntaxi, struktura byla uložena do druhé databáze se stejným pojmenováním tabulek a jejich entit.

Pro tvorbu indexu bylo použito příkazu:

ALTER TABLE název_tabulky ADD INDEX (požadovaný_sloupec);

Pomocí tohoto byly vytvořeny B-Tree indexy na následující kombinace tabulek a jejich sloupců. Tabulka person – id_zamestnani, rok_nar; Zajmy – nazev_zajem; Zajmy_lide – id_osoba, id_zajem.

4.4.4 Testovací dotazy pro systém eXist

Pro systém řízení báze dat eXist bylo na reprodukci dotazů v jazyce SQL využito kombinace funkcí jazyka XPath pro vyhledávání elementů a jazyka XQuery pro jejich agregaci. Testy 1,3 a 12 se nepodařilo přesně replikovat z důvodů odlišnosti jazyků SQL a XQuery. Jelikož se při dotazování využívá i jazyka XPath, kde je třeba definovat cestu do uloženého souboru a základní strukturu k požadovanému elementu, dotazy se liší, a proto budou uvedeny v pořadí, ve kterém probíhalo i měření. Jako první jsou uvedeny dotazy pro testování rozdělených dokumentů XML, které obsahují 10 000 záznamů v dokumentu „Person.XML“, poté budou uvedeny příkazy pro dotazování se nad spojenými dokumenty XML (10 000 a následně 100 000 záznamů), jako poslední jsou dotazy s vytvořenými indexy. Indexy byly vytvořeny nad spojeným dokumentem XML se 100 000 záznamy. Dotazy jsou označeny stejně jako dotazy v jazyce SQL, aby bylo snadnější se orientovat v porovnávání časů průběhu, také vracejí stejné hodnoty, jako jejich protějšky v jazyce SQL, tudíž není třeba znovu popisovat, jaké jsou výstupní hodnoty.

```
**Testy nad rozdělenými dokumenty**
**Test 2**
for $i in fn:doc("person10k.xml")/root/row
        where $i/rok_nar >= "2000-01-01"
        order by $i/rok_nar ascending
return <vysledek>{$i/first_name, $i/last_name, $i/rok_nar}</vysledek>

**Test 4**
for $i in fn:doc("person10k.xml")/root/row,
    $p in fn:doc("job10k.xml")/root/row[id_zamestnani = $i/id_zamestnani]
        where $p/nazev_zamestnani = "hasič"
return<vysledek>{$i/first_name, $i/last_name, $p/nazev_zamestnani}</vysledek>

**Test 5**
for $i in fn:doc("person10k.xml")/root/row,
    $p in fn:doc("job10k.xml")/root/row[id_zamestnani = $i/id_zamestnani]
        order by $p/nazev_zamestnani
return<vysledek>{$i/first_name, $i/last_name, $p/nazev_zamestnani}</vysledek>

**Test 6**
for $i in fn:doc("person10k.xml")/root/row,
    $p in fn:doc("job10k.xml")/root/row[id_zamestnani = $i/id_zamestnani]
        where $p/nazev_zamestnani = "hasič" and $i/rok_nar >= "1980-01-01"
return<vysledek>{$i/first_name, $i/last_name, $i/rok_nar}</vysledek>
```

Test 7

```
for $i in fn:doc("person10k.xml")/root/row,
    $p in fn:doc("job10k.xml")/root/row[id_zamestnani = $i/id_zamestnani]
return<vysledek>{$i/first_name, $i/last_name, $p/nazev_zamestnani}</vysledek>
```

Test 8

```
for $s in fn:doc("person10k.xml")/root/row
return<vysledek>{$s/first_name, $s/last_name,
        for $i in fn:doc("job10k.xml")/root/row [id_zamestnani = $s/id_zamestnani]
        return $i/nazev_zamestnani}</vysledek>
```

Test 9

```
for $i in fn:doc("person10k.xml")/root/row,
    $p in fn:doc("job10k.xml")/root/row[id_zamestnani = $i/id_zamestnani]
        where $p/nazev_zamestnani = "hasič"
return<vysledek pocet="{count($p/nazev_zamestnani)}"/>
```

Test 10

```
for $i in fn:doc("person10k.xml")/root/row,
    $p in fn:doc("zajmy_lide10k.xml")/root/row[id_osoba = $i/id_osoba],
    $s in fn:doc("zajmy10k.xml")/root/row[id_zajem = $p/id_zajem]
return<vysledek>{$s/nazev_zajem, $i/first_name, $i/last_name}</vysledek>
```

Test 11

```
for $i in fn:doc("person10k.xml")/root/row,
    $p in fn:doc("zajmy_lide10k.xml")/root/row[id_osoba = $i/id_osoba],
    $s in fn:doc("zajmy10k.xml")/root/row[id_zajem = $p/id_zajem]
where $s/nazev_zajem = "rybaření"
return<vysledek>{$s/nazev_zajem, $i/first_name, $i/last_name}</vysledek>
```

Testy nad jednotným dokumentem 10 000 záznamů

Test 2

```
for $i in fn:doc("emp10k.xml")/root/row
   where $i/rok_nar >= "2000-01-01"
   order by $i/rok_nar ascending
return<vysledek>{$i/first_name, $i/last_name, $i/rok_nar}</vysledek>
```

test 4

```
for $i in fn:doc("emp10k.xml")/root/row
   where $i/nazev_zamestnani = "hasič"
return<vysledek>{$i/first_name, $i/last_name, $i/nazev_zamestnani}</vysledek>
```

Test 5

```
for $i in fn:doc("emp10k.xml")/root/row
   order by $i/nazev_zamestnani
return<vysledek>{$i/first_name, $i/last_name, $i/nazev_zamestnani}</vysledek>
```

Test 6

```
for $i in fn:doc("emp10k.xml")/root/row
  where $i/nazev_zamestnani = "hasič" and $i/rok_nar >= "1980-01-01"
return<vysledek>{$i/first_name, $i/last_name, $i/rok_nar}</vysledek>
```

Test 7

```
for $i in fn:doc("emp100k.xml")/root/row
return<vysledek>{$i/first_name, $i/last_name, $i/nazev_zamestnani}</vysledek>
```

Test 8

Nad jednotným dokumentem vrací stejné hodnoty jako Test 7, data jsou v jednom dokumentu, není třeba dělat spojení.

Test 9

```
for $i in fn:doc("emp10k.xml")/root/row
   where $i/nazev_zamestnani = "hasič"
return<vysledek pocet="{count($i/nazev_zamestnani)}"/>
```

Test 10

```
for $i in fn:doc("emp10k.xml")/root/row
return
<vysledek>{$i/nazev_zajem, $i/first_name, $i/last_name}</vysledek>
```

Test 11

```
for $i in fn:doc("emp10k.xml")/root/row
   where $i/nazev_zajem = "rybaření"
return<vysledek>{$i/nazev_zajem, $i/first_name, $i/last_name}</vysledek>
```

Testy nad jednotným dokumentem 100 000 záznamů

Testy mají zcela shodnou strukturu, jediný rozdíl je změna v syntaxi XPath části dotazu, místo „fn:doc("emp10k.xml")/root/row“ bylo použito „fn:doc("emp100k.xml")/root/row“.

Testy s využitím indexů

Pro testování využití indexů v systému eXist a urychlení dotazů nad dokumenty byly zvoleny následující předpoklady. Testování bude provedeno nad XML dokumentem se 100 000 záznamy a bude použit rozsahový index. Nastavení indexů v databázovém systému eXist je dosaženo následujícím postupem. Nejprve je nutné v systémové kolekci databáze

„/db/system/config/“ definovat stejnou strukturu, jaká odpovídá umístění dat, které chceme indexovat. V našem případě bylo vytvořeno „/db/system/config/db/test“ a v této kolekci založit soubor „collection.xconf“. Do tohoto souboru je nutné uložit základní strukturu ve tvaru:

```
<collection xmlns="http://exist-db.org/collection-config/1.0">
<index><!—Definice indexů--></index>
</collection>
```

Mezi tagy index je možné definovat všechny typy indexů co systém řízení báze dat eXist nabízí. Pro testování bylo rozhodnuto použít rozsahové indexy, jelikož urychlují vyhledávání při využívání =, >, <, porovnání v XPath výrazech. Výsledná struktura nastavení indexů je následovná:

```
<collection xmlns="http://exist-db.org/collection-config/1.0">
   <index><create qname="nazev_zamestnani" type="xs:string"/>
      <create qname="nazev_zajem" type="xs:string"/>
      <create qname="rok_nar" type="xs:date"/></index>
</collection>
```

Pro použití indexu je třeba ještě upravit strukturu XQuery dotazu. Upravené dotazy, které nejvíce těžili, se zavedení rozsahového indexu jsou tyto:

```
**Test 2**
for $i in fn:doc("emp100k.xml")/root/row[rok_nar>='2000-01-01']
         order by $i/rok_nar ascending
return
      <vysledek>{$i/first_name, $i/last_name, $i/rok_nar}</vysledek>

**Test 4**
for $i in collection("/db/apps/test/emp100k.xml ")/root/row[nazev_zamestnani='hasič']
return
      <vysledek>{$i/first_name, $i/last_name, $i/nazev_zamestnani}</vysledek>

**Test 6**
for $i in fn:doc("emp100k.xml")/root/row[nazev_zamestnani='hasič', rok_nar>='1980-01-
01']
return
      <vysledek>{$i/first_name, $i/last_name, $i/rok_nar}</vysledek>

**Test 11**
    for $i in fn:doc("emp100k.xml")/root/row[nazev_zajem='rybaření']
return
      <vysledek>{$i/nazev_zajem, $i/first_name, $i/last_name}</vysledek>
```

4.4.5 Průběh testování

S připravenými daty a formulovanými dotazy je možné přistoupit k samotnému testování. Testy byly prováděny po sériích. Mezi sériemi byla časová prodleva kolem pěti minut. Výsledky testů byly zapisovány do tabulkového editoru Excel pro další analýzu. Výsledky testů prováděných v MySQL je možné číst ihned po provedení požadovaného dotazu (pokud bylo použito profilovacího výrazu), časy trvání v systému řízení báze dat eXist byly získány z logu, ten je umístěn v adresáři $EXIST_HOME/webapp/WEB-INF/logs/exist.log. Všechny testy byly celkově provedeny čtyřicetkrát. Po získání hodnot byla provedena základní statistická analýza pro určení měr polohy (průměr a medián) a variability (rozptyl, směrodatná odchylka a variační koeficient). Pro eliminaci odlehlých pozorování byl použit censorovaný průměr, ze souboru byly vyloučeny α procent nejvyšších a nejnižších hodnot a ze zbylých je spočten aritmetický průměr. Pro tento účel byla použita funkce v MS Excel TRIMMEAN a α byla zvolena na 9%. Tímto postupem byl vypočten průměr ze souboru, který je oproštěn o maximální a minimální hodnotu. Ostatní statistické ukazatele byly spočteny dle standardních postupů.

5. Výsledky a diskuze

Zde budou předvedeny výsledky testů, které byly definovány na konci předchozí kapitoly. Uvedeny budou statistické míry polohy a variability pro dané měření a grafické znázornění v podobě spojnicových grafů. Uveden je postup, jakým byly spočteny statistické hodnoty. Tyto spočtené hodnoty vycházejí z naměřených hodnot během testování. Všechny naměřené hodnoty jsou v elektronické verzi přiložené na CD. První jsou zmíněny výsledky všech testů na systému řízení báze dat MySQL a poté na systému eXist. Spojnicové grafy usnadňují interpretaci a přehlednost naměřených hodnot.

5.1 Výsledky měření pro systém MySQL

Prezentace výsledků měření získaných na systému řízení báze dat MySQL. Kompletní naměřené hodnoty z testu s 10 000 záznamy jsou obsaženy v příloze práce číslo 3. Pro svoji obsáhlost jsou zbylé hodnoty umístěny na přiložené CD. Z těchto hodnot byly určeny základní ukazatele míry polohy – censorovaný průměr a medián. Jelikož se spočtený průměr od mediánu v žádném měření výrazně nelišil, byl použit i pro výpočet ukazatelů variability – rozptylu, směrodatné odchylky a variačního koeficientu. Variační koeficient udává, jak moc jsou dat nesourodá, pokud se pohybuje v rozsahu -0,5 – +0,5 hodnoty se pohybují v blízkosti střední hodnoty. Směrodatná odchylka vyjadřuje, že více než 50% naměřených hodnot se neodchyluje od průměru v obou směrech o hodnotu směrodatné odchylky. Hodnoty průměru, mediánu a směrodatné odchylky jsou v sekundách. Rozptyl je v sekundách^2 a variační koeficient je bezrozměrný ukazatel. Čím menší hodnoty, tím lepší.

	Míry polohy		Míry variability		
	$\bar{X}$	$\tilde{X}$	s^2	s	V
Test 1	0,013206	0,012385	0,000015	0,003887	0,294353
Test 2	0,006149	0,006201	0,000002	0,001486	0,241694
Test 3	0,031248	0,028305	0,000095	0,009734	0,311494
Test 4	0,011398	0,009936	0,000015	0,00383	0,336037
Test 5	0,154907	0,152441	0,000657	0,025638	0,165508
Test 6	0,010036	0,008926	0,000008	0,002915	0,290429
Test 7	0,150938	0,146109	0,001603	0,040032	0,26522
Test 8	0,153926	0,145468	0,000429	0,020716	0,134586
Test 9	0,009107	0,008062	0,000011	0,003339	0,366616
Test 10	0,017328	0,018625	0,000031	0,005578	0,321917
Test 11	0,005462	0,004994	0,000001	0,001172	0,214665
Test 12	0,010106	0,008839	0,000011	0,003279	0,32448

Tabulka 5-1 výsledky měření MySQL soubor s 10 000 záznamy

	Míry polohy		Míry variability		
	$\bar{X}$	$\tilde{X}$	s^2	s	V
Test 1	0,016286	0,015182	0,000028	0,005332	0,327376
Test 2	0,003866	0,003221	0,000002	0,001444	0,373526
Test 3	0,032932	0,033719	0,000072	0,008513	0,258490
Test 4	0,001690	0,001755	0,000000	0,000406	0,240220
Test 5	0,013803	0,013939	0,000001	0,000806	0,058383
Test 6	0,002662	0,002712	0,000004	0,001873	0,703407
Test 7	0,014755	0,015244	0,000008	0,002803	0,189959
Test 8	0,160536	0,161952	0,000220	0,014818	0,092301
Test 9	0,000413	0,000414	0,000000	0,000022	0,053893
Test 10	0,012386	0,012337	0,000006	0,002381	0,192208
Test 11	0,002453	0,002472	0,000000	0,000399	0,162590
Test 12	0,009118	0,009197	0,000000	0,000464	0,050861

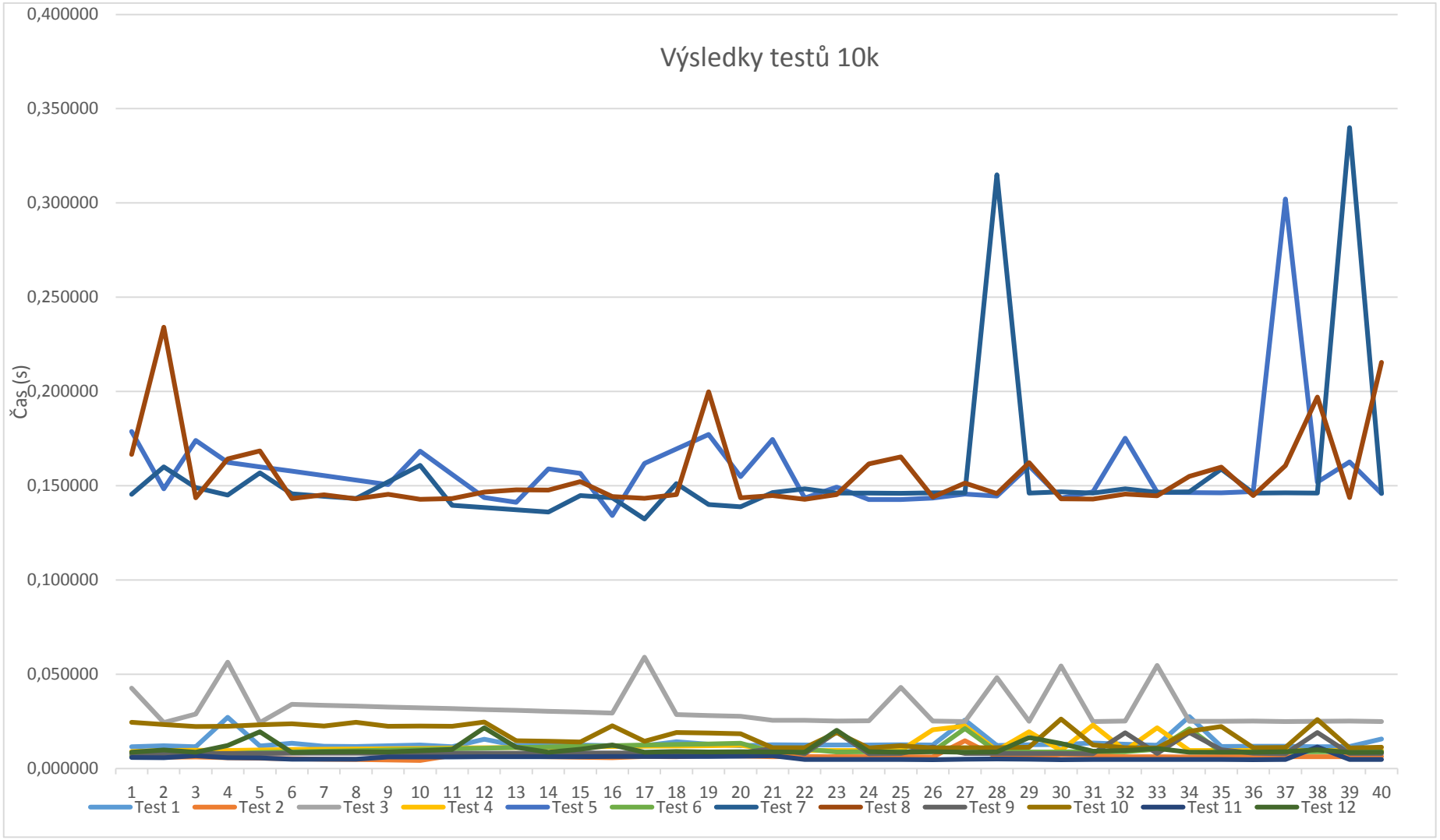
Tabulka 5-2 výsledky měření MySQL soubor s 10 000 záznamy, použity indexy

	Míry polohy		Míry variability		
	$\bar{X}$	$\tilde{X}$	s^2	s	V
Test 1	0,168206	0,164819	0,0002654	0,016292	0,096859
Test 2	0,06665	0,066931	0,0004494	0,021199	0,318067
Test 3	0,265026	0,26137	0,0007052	0,026555	0,100198
Test 4	0,099159	0,098196	0,0001154	0,010743	0,108336
Test 5	1,402515	1,380746	0,0022493	0,047427	0,033816
Test 6	0,094382	0,090987	0,000674	0,025962	0,275079
Test 7	1,466588	1,423397	0,0139324	0,118036	0,080483
Test 8	1,495714	1,510682	0,0046423	0,068135	0,045553
Test 9	0,085772	0,085722	9,236E-05	0,009611	0,112047
Test 10	0,042536	0,04387	3,624E-05	0,00602	0,141529
Test 11	0,019689	0,014919	6,647E-05	0,008153	0,414064
Test 12	0,062591	0,054202	0,0004494	0,021198	0,338672

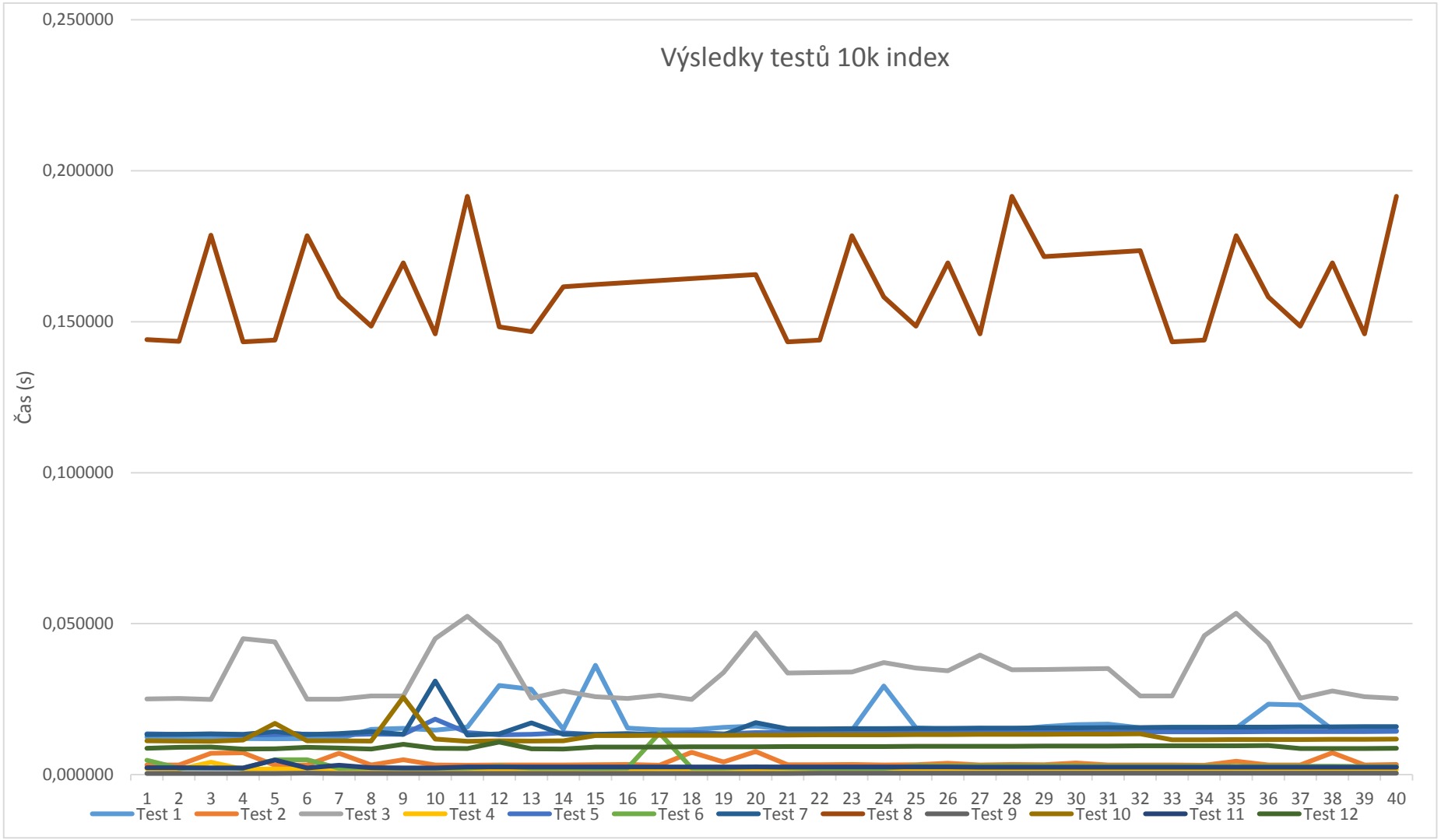
Tabulka 5-3 výsledky měření MySQL soubor s 100 000 záznamy

	Míry polohy		Míry variability		
	$\bar{X}$	$\tilde{X}$	s^2	s	V
Test 1	0,160045	0,156778	0,001373	0,037054	0,23152
Test 2	0,033325	0,033466	0,000112	0,0106	0,318067
Test 3	0,27402	0,262958	0,001147	0,033873	0,123614
Test 4	0,014717	0,012947	6,12E-05	0,007823	0,531542
Test 5	0,140703	0,142136	0,000129	0,011363	0,08076
Test 6	0,017124	0,016143	2,25E-05	0,004744	0,277055
Test 7	0,132828	0,130224	0,000178	0,013345	0,100465
Test 8	1,400832	1,400802	0,001847	0,042982	0,030683
Test 9	0,002401	0,002392	2,18E-07	0,000467	0,194433
Test 10	0,042764	0,039502	0,000138	0,011743	0,274598
Test 11	0,002704	0,002423	7,7E-07	0,000877	0,32441
Test 12	0,044141	0,045737	0,000141	0,01188	0,269145

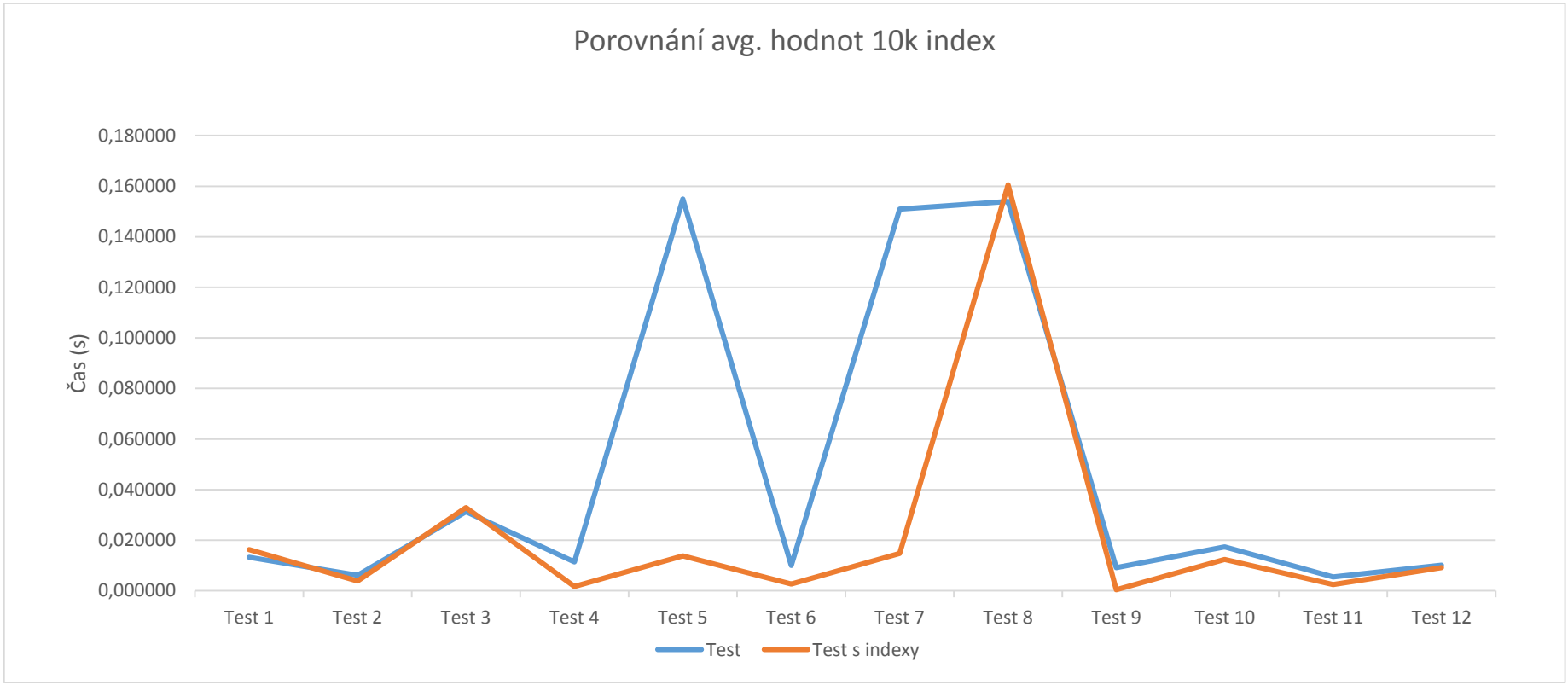
Tabulka 5-4 výsledky měření MySQL soubor s 100 000 záznamy, použity indexy



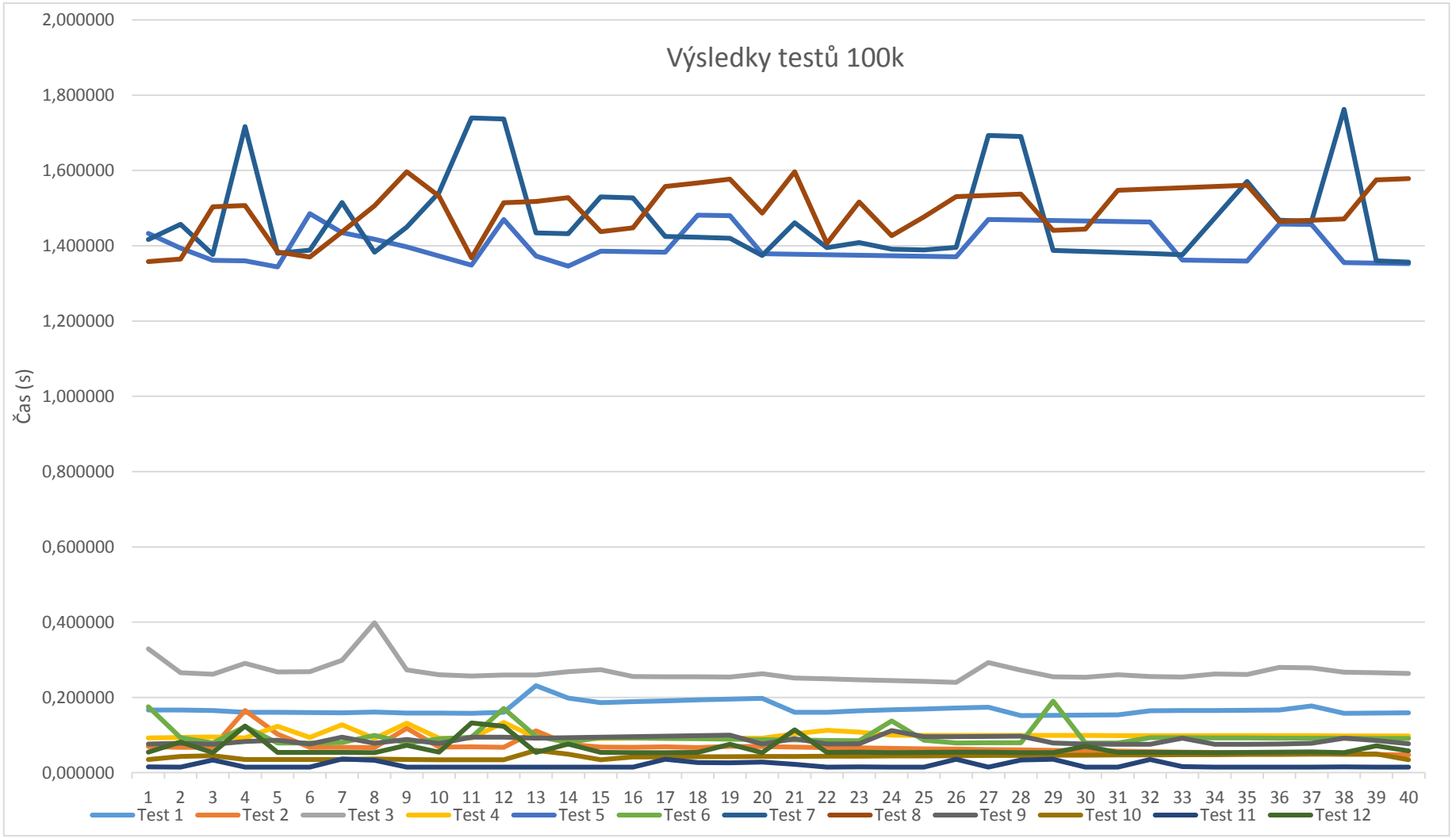
Graf 5-1 srovnání výsledků měření MySQL soubor s 10 000 záznamy



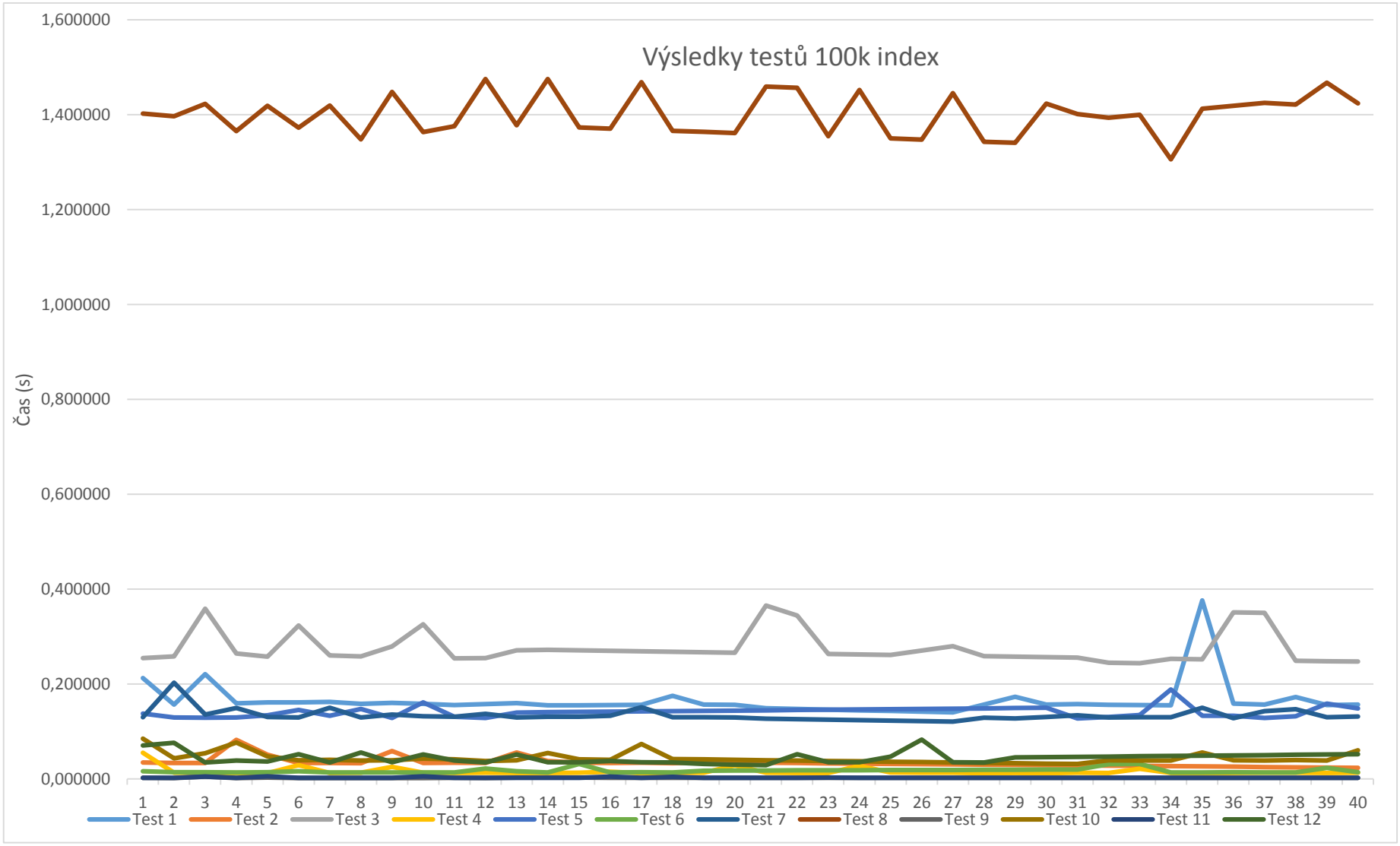
Graf 5-2 srovnání výsledků měření MySQL soubor s 10 000 záznamy, použity indexy



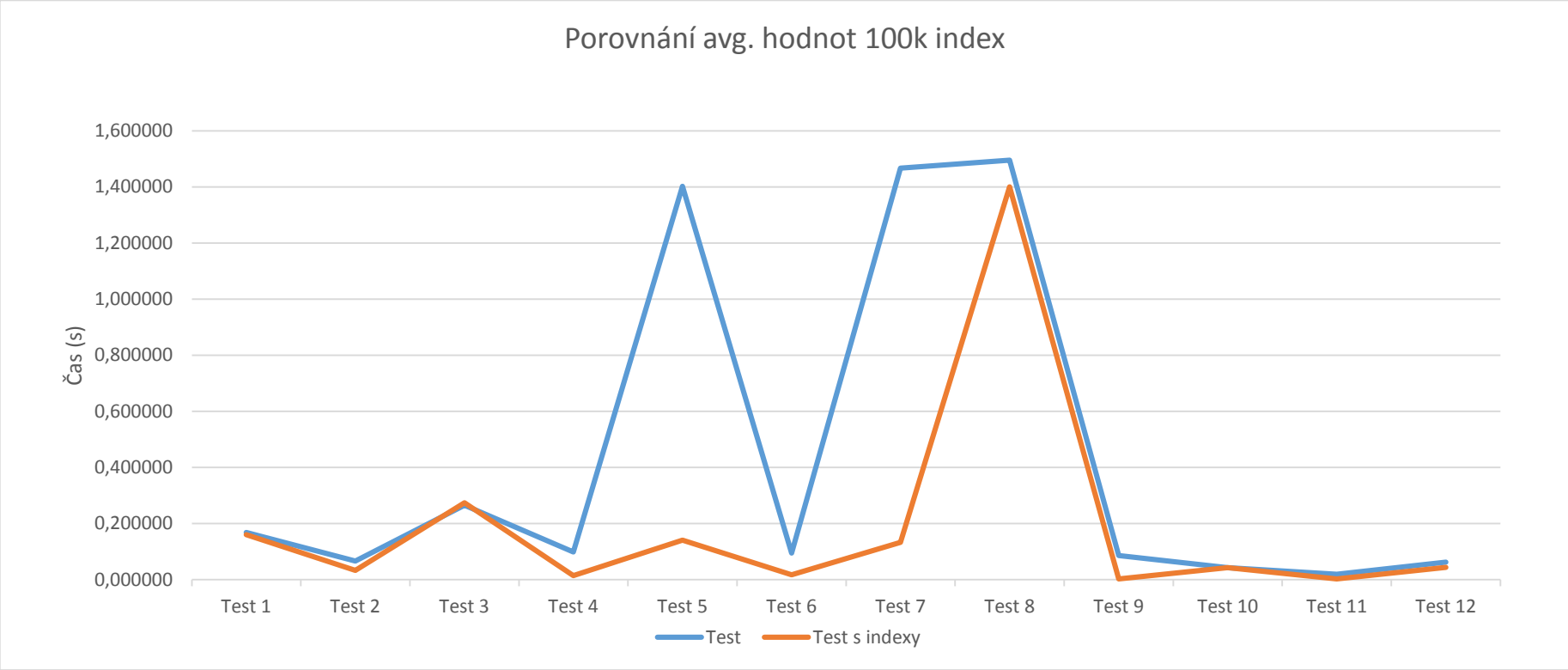
Graf 5-3 srovnání výsledků měření MySQL průměrné hodnoty bez a s použitím indexace



Graf 5-4 srovnání výsledků měření MySQL soubor s 100 000 záznamy


Výsledky testů 100k index
Čas (s)
1,600000
1,400000
1,200000
1,000000
0,800000
0,600000
0,400000
0,200000
0,000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Test 1
Test 2
Test 3
Test 4
Test 5
Test 6
Test 7
Test 8
Test 9
Test 10
Test 11
Test 12

Graf 5-5 srovnání výsledků měření MySQL soubor s 100 000 záznamy, použity indexy



Graf 5-6 srovnání výsledků měření MySQL průměrné hodnoty bez a s použitím indexace

5.2 Výsledky měření pro systém eXist

Výsledky měření získaných na systému řízení báze dat eXist. Kompletní seznam všech naměřených hodnot je přiložen na CD. Uvedeny jsou hodnoty po provedení základní statistické analýzy. Oproti testování systému MySQL došlo k několika úpravám, testy 1,3 a 12 jsou vynechány z důvodů odlišné syntaxe jazyka XQuery. Testování proběhlo nejdříve na oddělených dokumentech, které byly pro potřeby dotazů spojovány pomocí klíčování. Ačkoli syntaxe jazyku XQuery toto umožňuje, časy těchto dotazů byly natolik vysoké, že bylo rozhodnuto tímto způsobem testovat pouze data obsahující 10 000 záznamů v dokumentu „person.xml“. Následné testování se soustředilo na testování dokumentů obsahující všechny data z původních tabulek. V případě testování těchto spojených dokumentů bylo vyzkoušeno i indexování a to na dokumentu se 100 000 záznamy. Hodnoty průměru, mediánu a směrodatné odchylky jsou v sekundách. Rozptyl je v sekundách^2 a variační koeficient je bezrozměrný ukazatel.

	Míry polohy			Míry variability		
	$\bar{X}$	$\tilde{X}$		s^2	s	V
Test 2	0,081134	0,072136		0,000666	0,025815	0,318182
Test 4	0,978897	0,966004		0,003573	0,059772	0,061061
Test 5	1,137553	1,132065		0,000952	0,030847	0,027117
Test 6	0,948867	0,95514		0,00184	0,042894	0,045205
Test 7	1,013158	1,012883		0,000186	0,013646	0,013469
Test 8	1,144769	1,140765		0,005364	0,07324	0,063978
Test 9	0,986517	0,987339		0,002683	0,051802	0,05251
Test 10	376,7779	377,2276		3,460133	1,860143	0,004937
Test 11	402,5331	403,1611		6,730211	2,594265	0,006445

Tabulka 5-5 výsledky měření eXist soubor s 10 000 záznamy, oddělené dokumenty

	Míry polohy			Míry variability		
	$\bar{X}$	$\tilde{X}$		s^2	s	V
Test 2	0,10446	0,091284		0,000916	0,030271	0,289789
Test 4	0,054196	0,04805		0,000259	0,016097	0,297009
Test 5	0,231951	0,230348		0,000301	0,017339	0,074753
Test 6	0,105677	0,103206		0,000406	0,020159	0,190761
Test 7	0,178392	0,173429		0,001165	0,034135	0,191349
Test 9	0,050358	0,044717		0,000242	0,015552	0,308841
Test 10	0,173031	0,158676		0,001232	0,035102	0,202865
Test 11	0,03827	0,035049		0,00011	0,010488	0,274061

Tabulka 5-6 výsledky měření eXist soubor s 10 000 záznamy jeden dokument

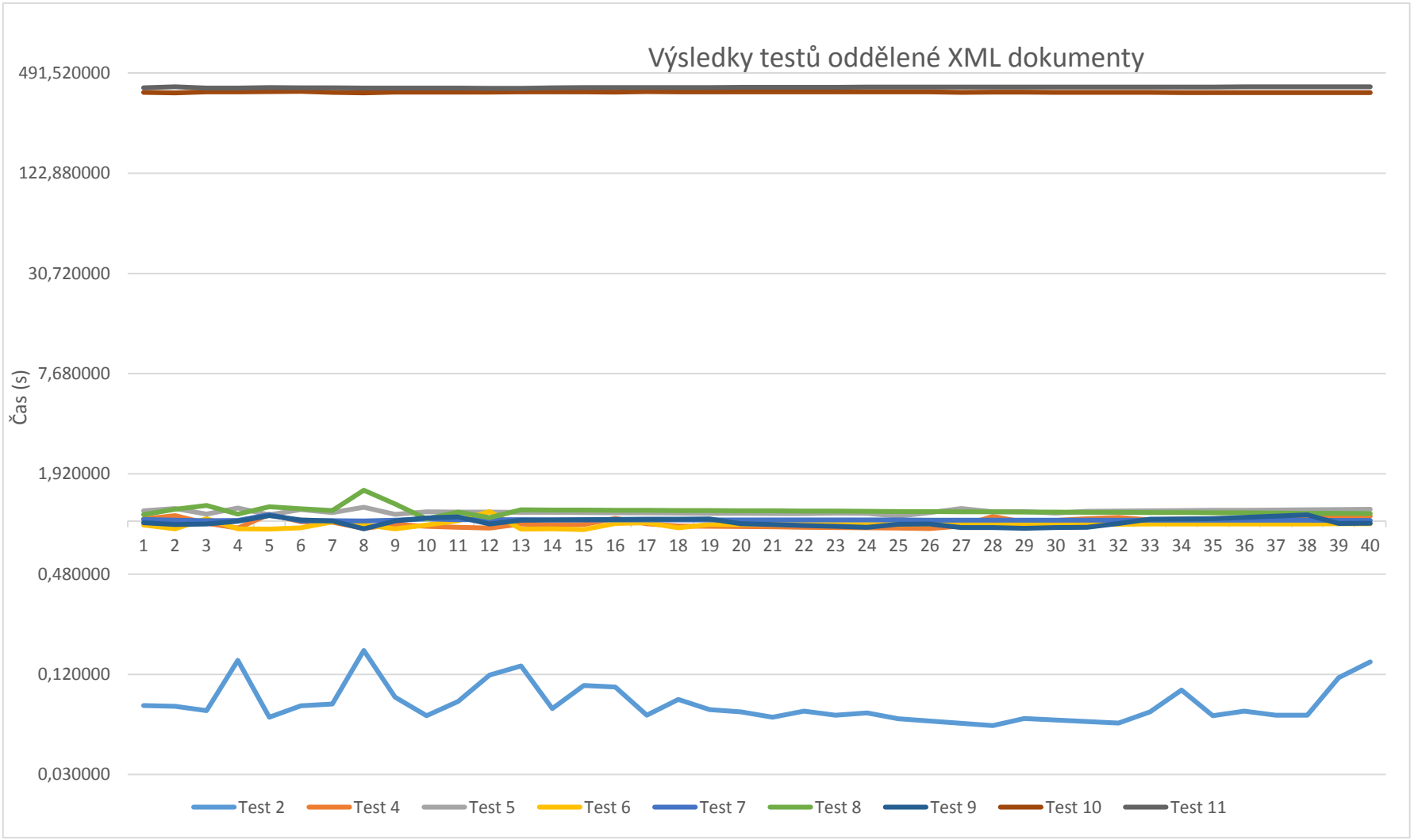
	Míry polohy			Míry variability	
	$\bar{X}$	$\tilde{X}$	s^2	s	V
Test 2	0,770534	0,772845	0,00959	0,097931	0,127094
Test 4	0,474654	0,442565	0,005749	0,075824	0,159745
Test 5	2,430919	2,431698	0,014696	0,121228	0,049869
Test 6	0,842523	0,830101	0,002251	0,047445	0,056313
Test 7	1,616884	1,6108	0,004011	0,063334	0,03917
Test 9	0,381168	0,3785	0,001738	0,041694	0,109384
Test 10	1,632427	1,623255	0,04315	0,207726	0,12725
Test 11	0,289136	0,288043	0,001234	0,035127	0,121488

Tabulka 5-7 výsledky měření eXist soubor s 100 000 záznamy jeden dokument

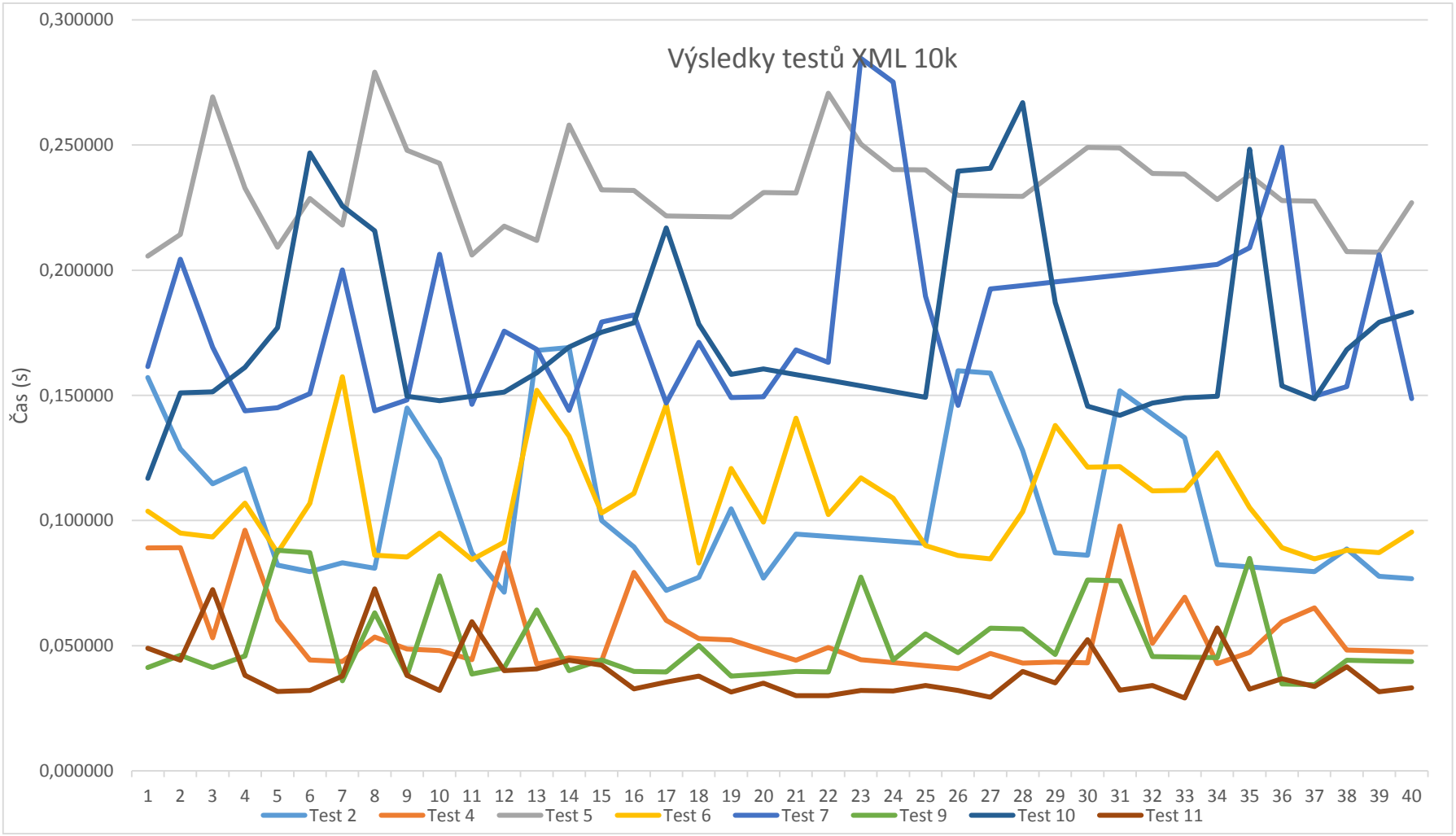
	Míry polohy		Míry variability		
	$\bar{X}$	$\tilde{X}$	s^2	s	V
Test 2	0,47318	0,473942	0,002765	0,052587	0,111134
Test 4	0,280598	0,272364	0,001676	0,040939	0,1459
Test 6	0,451393	0,44216	0,000579	0,024057	0,053295
Test 11	0,148506	0,145473	0,000321	0,017921	0,120679

Tabulka 5-8 výsledky měření eXist soubor s 100 000 záznamy jeden dokument, indexy

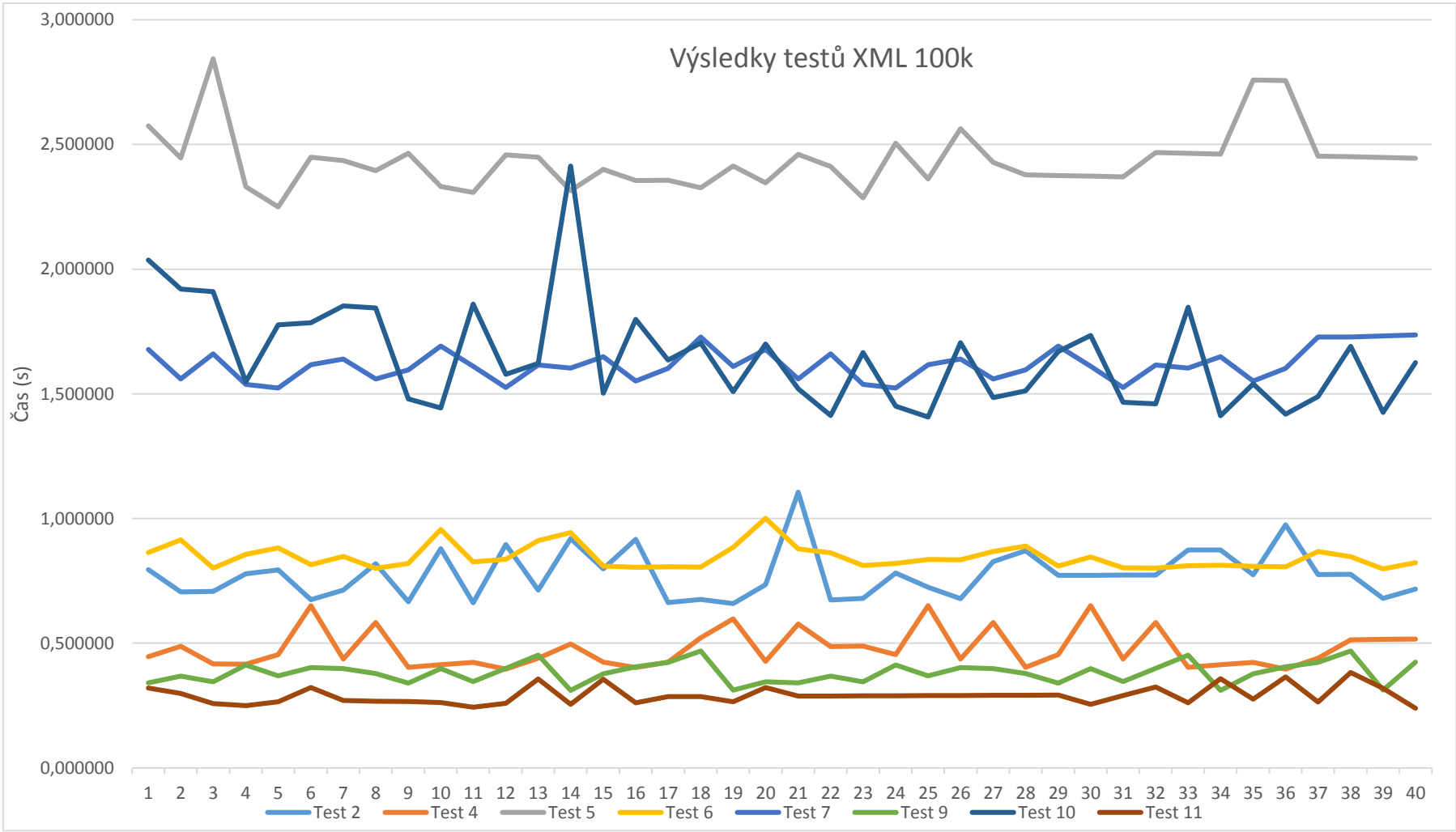
Grafické znázornění získaných hodnot měření na systému eXist je na následujících grafech. Na grafu 5-7 bylo třeba nastavit logaritmické měřítko na ose x z důvodů velkých rozsahů trvání testů – dotazy z testů 10 a 11 trvaly kolem 6 minut každý. Z tohoto důvodu nebyla testována možnost více dokumentů se 100 000 záznamy. Zbylé grafy mají lineární osy. Graf porovnává rozdíly v trvání dotazů po nadefinování rozsahových indexů nad testovanou kolekcí dokumentů. Měření proběhlo jen s dotazy, které tyto indexy dokáží využít.



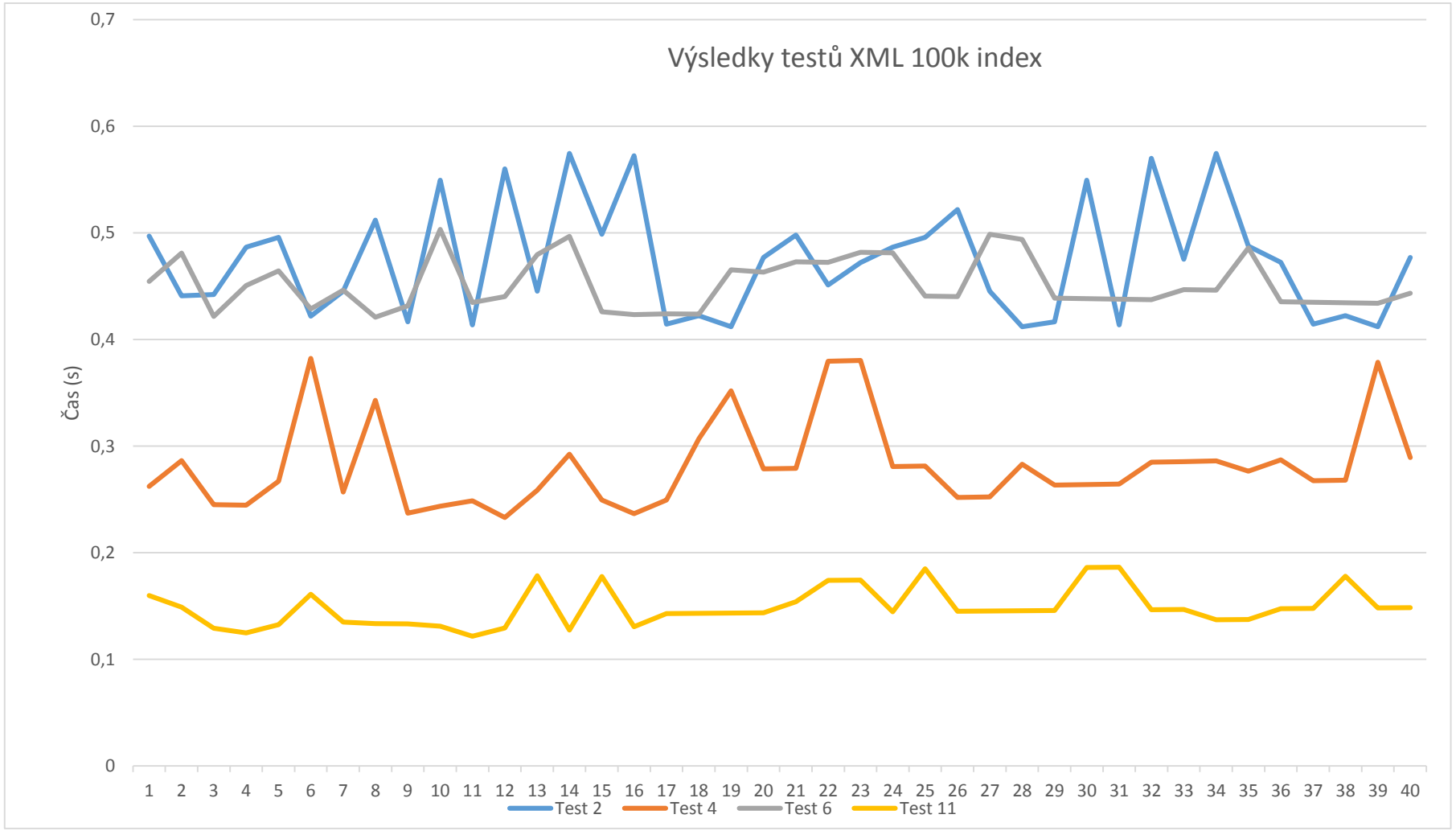
Graf 5-7 srovnání výsledků měření eXist oddělené XML dokumenty, 10 000 záznamů



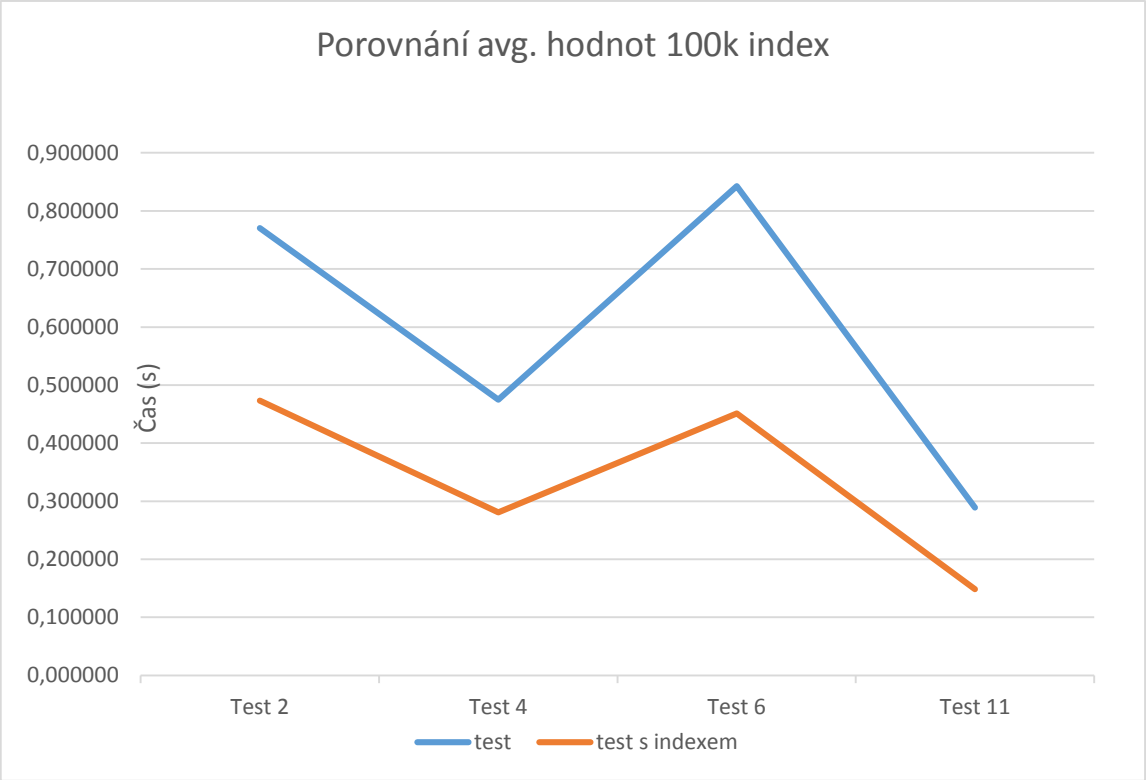
Graf 5-8 srovnání výsledků měření eXist jeden XML dokument, 10 000 záznamů



Graf 5-9 srovnání výsledků měření eXist jeden XML dokument, 100 000 záznamů



Graf 5-10 srovnání výsledků měření eXist jeden XML dokument, 100 000 záznamů, indexy



Graf 5-11 srovnání výsledků měření eXist průměrné hodnoty bez a s použitím indexace

5.3 Porovnání výsledků

Pro vzájemné porovnání byly vybrány hodnoty testů naměřené na datové základně s 10 000 záznamy bez využití indexace a tety provedené na podkladech s 100 000 záznamy nejprve bez definovaných indexů a následně pro porovnání s aktivní indexací. Porovnávány jsou spočtené censorované průměrné hodnoty, které jsou uvedeny v následujících tabulkách.

	MySQL		eXist
	$\bar{X}$		$\bar{X}$
Test 2	0,006149		0,081134
Test 4	0,011398		0,978897
Test 5	0,154907		1,137553
Test 6	0,010036		0,948867
Test 7	0,150938		1,013158
Test 8	0,153926		1,144769
Test 9	0,009107		0,986517
Test 10	0,017328		376,7779
Test 11	0,005462		402,5331

	MySQL		eXist
	$\bar{X}$		$\bar{X}$
Test 2	0,006149		0,104460
Test 4	0,011398		0,054196
Test 5	0,154907		0,231951
Test 6	0,010036		0,105677
Test 7	0,150938		0,178392
Test 9	0,009107		0,050358
Test 10	0,017328		0,173031
Test 11	0,005462		0,038270

Tabulka 5-9 porovnání průměrných hodnot 10k bez indexů

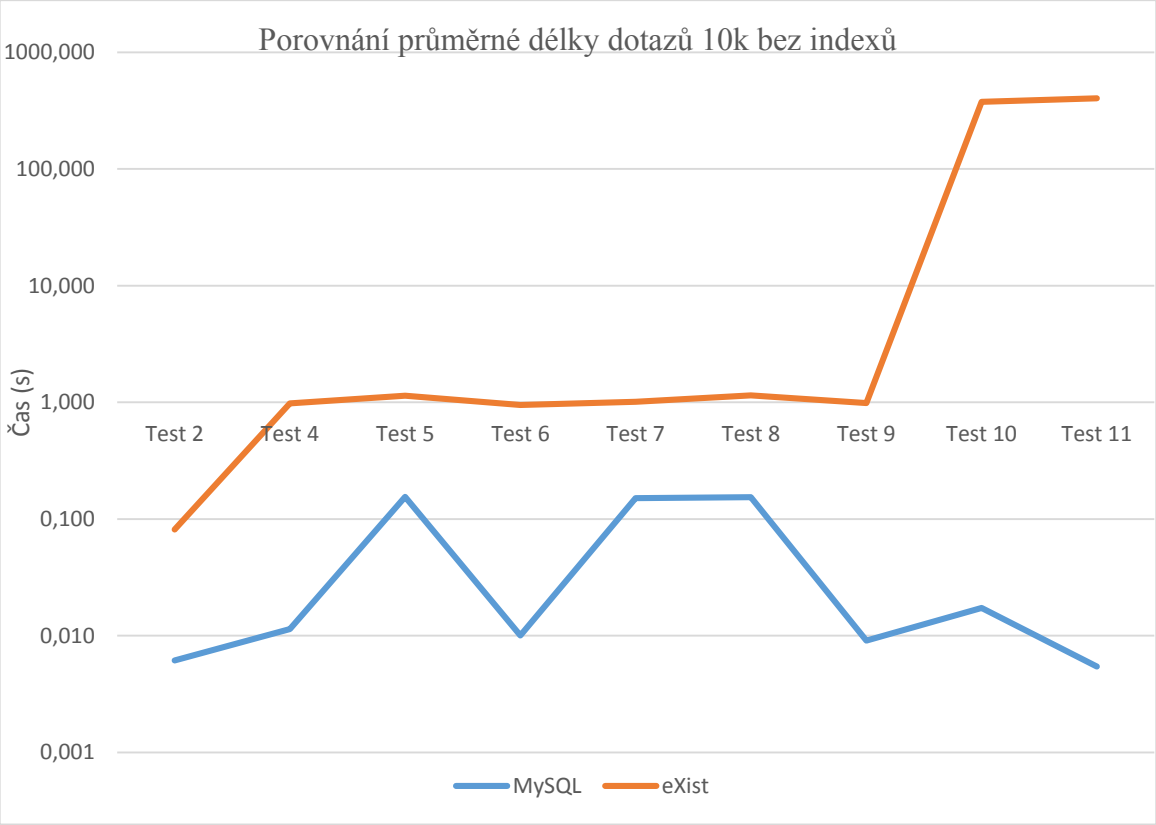
Tabulky 5-9 porovnávají hodnoty z testů, které byly provedeny na podkladech s 10 000 záznamy bez nastavené indexace. První tabulka zachycuje testování s rozdělenými daty do více XML dokumentů, zatímco druhá tabulka obsahuje hodnoty naměřené na jednom sloučeném XML dokumentu. Grafické vyjádření je na grafech 5-12 a 5-13.

	MySQL		eXist
	$\bar{X}$		$\bar{X}$
Test 2	0,06665		0,770534
Test 4	0,099159		0,474654
Test 5	1,402515		2,430919
Test 6	0,094382		0,842523
Test 7	1,466588		1,616884
Test 9	0,085772		0,381168
Test 10	0,042536		1,632427
Test 11	0,019689		0,289136

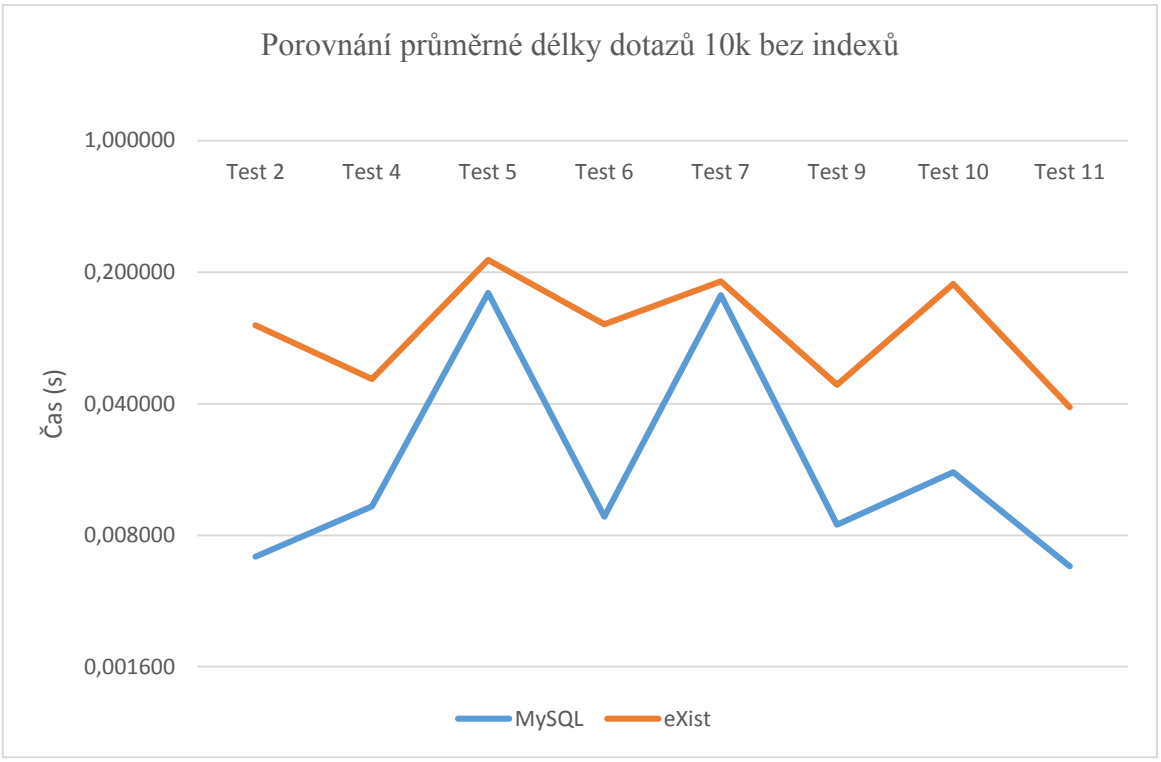
	MySQL		eXist
	$\bar{X}$		$\bar{X}$
Test 2	0,033325		0,47318
Test 4	0,014717		0,280598
Test 6	0,017124		0,451393
Test 11	0,002704		0,148506
Test 11	0,033325		0,47318

Tabulka 5-10 porovnání průměrných hodnot 100k bez a s indexy

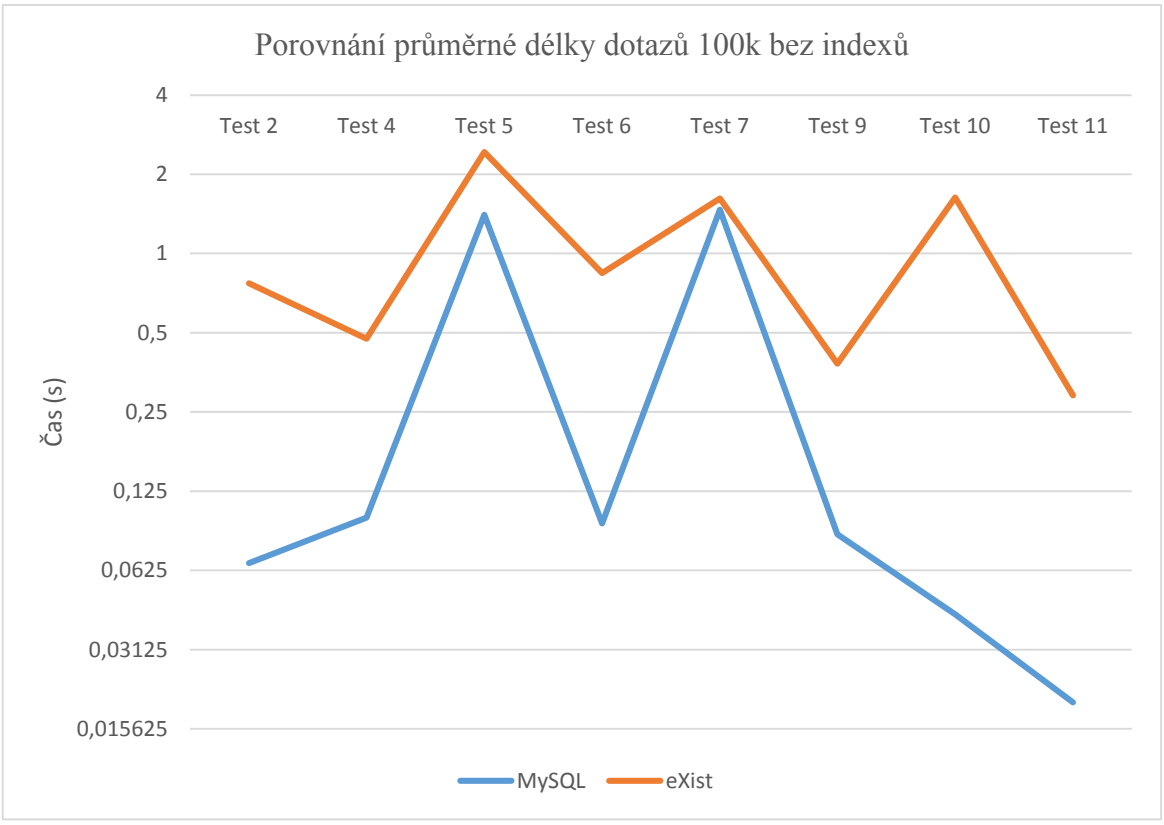
Tabulky 5-10 porovnávají hodnoty testů provedených na podkladech s 100 000 záznamy. První tabulka jsou hodnoty bez využití indexů, druhá s indexy. Grafické vyjádření je na grafech 5-14 a 5-15.



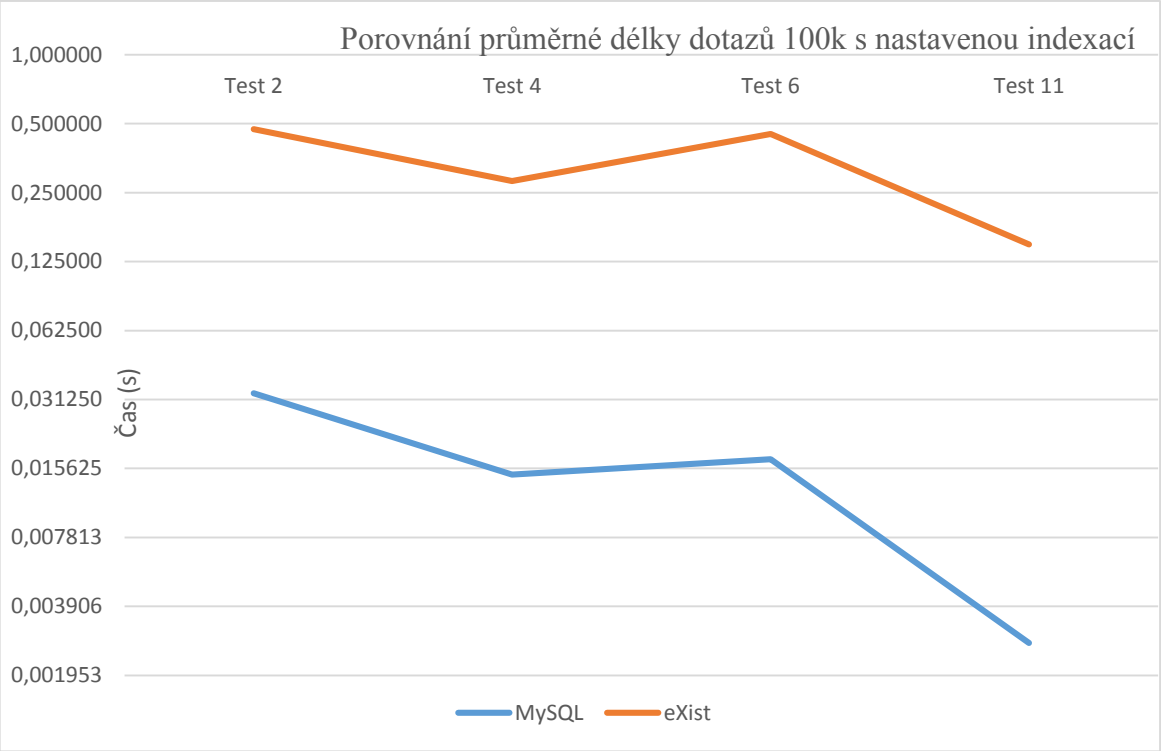
Graf 5-12 porovnání délky dotazů 10k bez indexů oddělené XML dokumenty



Graf 5-13 porovnání délky dotazů 10k bez indexů jeden XML dokument



Graf 5-14 porovnání délky dotazů 100k bez indexů



Graf 5-15 porovnání délky dotazů 100k s indexy

Z výsledků testů je jasně patrná rychlost systému řízení báze dat MySQL oproti systému eXist. Ve většině testů MySQL dosahovalo až o jeden řád rychlejší hodnoty než byly naměřeny na systému eXist. Systém indexace je u obou systémů velmi účinný a dokázal při správném nastavení urychlit vykonávání dotazů. Ačkoliv systém eXist nedosáhl rychlostně na systém MySQL nedá se tvrdit, že jde o spatný systém řízení báze dat, jen se prokázalo, že nedokáže zpracovávat datově orientované dokumenty tak rychle jako model relační databáze. Systém eXist vychází z dokumentově orientovaného modelu a ten je primárně určen pro navigaci v silně hierarchické struktuře dat. Pod tímto pojmem si lze představit například celou knihu uloženou ve formátu XML. Součástí rozšířené instalace systému eXist je i kompletní dílo Williama Shakespeara uložené ve formátu XML, pomocí následujícího příkazu XQuery je možné prohledat uložené dokumenty a zobrazit potřebný výsledek.

```
for $line in collection('/db/apps/exist101/data')//SPEECH/LINE[contains(., 'To be, or not to be')] return
<LINE play="{base-uri($line)}">{string($line)}</LINE>
```

```
<LINE play="/db/apps/exist101/data/hamlet.xml">
To be, or not to be: that is the question:</LINE>
```

Dotaz vrátí název díla a celý řádek jeho výskytu. Pro dosažení podobného výsledku v systému MySQL by bylo třeba velmi složité a pracné transformace celého dokumentu do soustavy tabulek, které by vyjadřovali jeho hierarchii pomocí cizích klíčů. Systém eXist tento dotaz zvládl pod 1 sekundu a umožňuje dále modifikovat dotazy a výsledek zobrazit jako webovou stránku.

Volba mezi systémy řízení báze dat MySQL a eXist ve výsledku především záleží na typu dat v datové základně, které jsou třeba spravovat. Pokud je třeba efektně spravovat ne hierarchické datové podklady v lepším případě s fixní strukturou pro většinu záznamů, zvolíme systém řízení báze dat vycházející z relačního modelu jako je MySQL. Pokud ale máme data hierarchická s danou strukturou, dokumentově orientovaný systém eXist může být to pravé. Jelikož první možnost typu dat je v dnešní době velmi rozšířená napřič různými informačními technologiemi je i relační model systému řízení báze dat velmi rozšířeným koncovým řešením. Využití pro dokumentově orientovaný model využívající nativní XML databázi eXist jsou velmi specifická a výsledná aplikace bývá často už od začátku vývoje profilována tímto směrem.

Budoucí vývoj moc nativním XML databázovým systémům nepřeje, většina hlavních vývojářů na poli relačních databází (Oracle, IBM, Microsoft) představila podporu pro ukládání XML dokumentů, MySQL tuto funkci přidal ve verzi 5.1.5. Společnost IBM dokonce svůj produkt DB2 rozšířilo o plnou podporu ukládání a i dotazováni nad XML dokumenty za pomocí XPath a XQuery jazyků a jejich systém řízení báze dat bývá místo označován jako hybridní místo původního relačního. Toto řešení koncovému zákazníkovi poskytuje klady z obou modelů, systém DB2 od IBM je ale komerční řešení. Na poli open source systémů si nativní XML databáze své místo nejspíše udrží kvůli oblíbenosti ve svých komunitách, které tyto systémy nadále vyvíjejí a poskytují jim i podporu pro nové uživatele.

6. Závěr

Hlavním cílem práce bylo porovnat klady a zápory dvou systémů řízení báze dat, systému MySQL založeného na relačním modelu a dokumentově orientovaného databázového systému využívající značkovací jazyk XML eXist. Tohoto cíle bylo dosaženo nejdříve popisem základních teoretických východisek a základního popisu obou systémů. Zmíněna byla architektura, historie, licencování a základní funkce. Dále je zmíněna základní problematika výkonnostního testování systémů řízení báze dat. Základní osvědčené postupy a několik softwarových nástroj, které se často pro výkonnostní testování využívají. V rámci praktického srovnání databázových systémů MySQL a eXist je následně navržena jednoduchá metodika testování na základě měření délky trvání dotazů nad určitou datovou strukturou. Jelikož se nepodařilo získat reálné datové podklady pro účely testování, byla navržena a z náhodných hodnot vygenerována testovací struktura ve dvou provedení, první menší obsahuje 10 000 záznamů a druhá větší se 100 000 záznamy. Dále byly navrženy testovací dotazy v jazyce SQL, které pokrývají většinu základních funkcí pro dotazování (projekce, selekce, spojení). Pomocí těchto dotazů byla otestován systém MySQL pracující nad vygenerovanými datovými podklady. Pro potřeby provedení podobných testů na systému eXist bylo třeba upravit datové podklady do formy dokumentů XML a vyjádřit dotazy v jazyce SQL pomocí kombinace jazyků XPath a XQuery. Toho se povedlo docílit na více než polovině testovacích dotazů. Výsledky získané z těchto testů byly podrobeny základní statistické analýze pro snadnější porovnání a interpretaci. Využito bylo i grafického znázornění pro lepší přehled a vzájemné porovnání. V rámci tohoto testování byla ověřena funkce základních indexů v obou systémech řízení báze dat. V rámci porovnání výsledků testování jsou zmíněny silné a slabé stránky testovaných systémů a je nastíněn možný budoucí vývoj pro zmíněné systémy řízení báze dat.

7. Seznam použitých zdrojů:

1. FAWCETT, Joe, QUIN, R. E. Liam, AYERS, Danny. *Beginning XML*. 5th Edition. John Wiley & Sons, Inc., 2012. 868 s. ISBN: 978-1-118-16213-2.

2. HOSOYA, Haruo. *Fundations of XML processing.* United States of America by Cambridge University Press, 2011. 240 s. ISBN: 978-0-511-90402-8.

3. SCHWARTZ, Baron, ZAITSEV, Peter, TKACHENKO, Vadim. *High Performance MySQL*. 3rd Edition. O'Reilly Media, 2012. 826 s. ISBN: 978-1-449-31428-6

4. DUBOIS, Paul. *MySQL Cookbook*. 3rd Edition. O'Reilly Media, 2014. 866 s. ISBN: 978-1-449-37402-0

5. DUBOIS, Paul. *MySQL*. 5th edition. Pearson Education Inc., 2013. 1176 s. ISBN: 978-0321833877.

6. HOSKINS, Dorothy. *XML and InDesign.* 1st Edition. O'Reilly Media, 2013. 148 s. ISBN: 978-1449344160

7. DAVIES, Alex. *High Availability MySQL Cookbook*. Pack Publishing Ltd., 2010. 276 s. ISBN: 978-1-847199-94-2.

8. ARENAS, Marcelo, BARCELO, Pablo, LIBKIN, Leonid, MURLAK, Filip. *Relational and XML Data Exchange*. Morgan & Claypool Publishers, 2010. 106 s. ISBN: 9781608454129.

9. POWELL, Gavin. *Beginning XML database.* John Wiley & Sons, Inc., 2007. 499 s. ISBN-13: 978-0-471-79120-1.

10. KOFLER, Michael. *Mistrovství v MySQL 5.* Computer Press, a.s., 2007. 808 s. ISBN: 978-80-251-1502-2.

11. RUSSELL, Dyer. *MySQL in a Nutshell*. 2nd Edition. O´Reilly Media Inc., 2008. 566 s. ISBN: 978-0-596-51433-4.

12. OPPEL, Andy. *SQL bez předchozích znalostí*. Computer Press, a.s., 2008. 242 s. ISBN: 978-80-251-1707-1.

13. SIEGEL, Erik, RETTER, Adam. *eXist*. O´Reilly Media Inc., 2014. 581 s. ISBN: 978-1-449-33710-0

8. Seznam obrázků:

9. Seznam tabulek:

10. Seznam grafů:

11. Přílohy:

1. Příklad dokumentově orientovaného XML dokumentu (popis výrobku):

```
<Product>
 <Intro>
 The <ProductName>Turkey Wrench</ProductName> from <Developer>Full
 Fabrication Labs, Inc.</Developer> is <Summary>like a monkey wrench,
 but not as big.</Summary>
 </Intro>
 <Description>
 <Para>The turkey wrench, which comes in <i>both right- and left-
 handed versions (skyhook optional)</i>, is made of the <b>finest
```

```
stainless steel</b>. The Readi-grip rubberized handle quickly adapts
to your hands, even in the greasiest situations. Adjustment is
possible through a variety of custom dials.</Para>
<Para>You can:</Para>
<List>
<Item><Link URL="Order.html">Order your own turkey wrench</Link></Item>
<Item><Link URL="Wrenches.htm">Read more about wrenches</Link></Item>
<Item><Link URL="Catalog.zip">Download the catalog</Link></Item>
</List>
<Para>The turkey wrench costs <b>just $19.99</b> and, if you
order now, comes with a <b>hand-crafted shrimp hammer</b> as a
bonus gift.</Para>
  </Description>
</Product>
```

2. Příklad datově orientovaného XML dokumentu (prodejní objednávka):

```
<SalesOrder SONumber="12345">
    <Customer CustNumber="543">
      <CustName>ABC Industries</CustName>
      <Street>123 Main St.</Street>
      <City>Chicago</City>
      <State>IL</State>
      <PostCode>60609</PostCode>
    </Customer>
    <OrderDate>981215</OrderDate>
    <Item ItemNumber="1">
      <Part PartNumber="123">
        <Description>
          <p><b>Turkey wrench:</b><br />
          Stainless steel, one-piece construction,
          lifetime guarantee.</p>
        </Description>
        <Price>9.95</Price>
      </Part>
      <Quantity>10</Quantity>
    </Item>
    <Item ItemNumber="2">
      <Part PartNumber="456">
        <Description>
          <p><b>Stuffing separator:<b><br />
          Aluminum, one-year guarantee.</p>
        </Description>
        <Price>13.27</Price>
      </Part>
      <Quantity>5</Quantity>
    </Item>
  </SalesOrder>
```

3. Naměřené hodnoty délky trvání dotazů v systému MySQL s 10 000 záznamy bez a s využitím indexů (hodnoty jsou v sekundách)

Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Test 11	Test 12
0,011527	0,006636	0,042654	0,009052	0,178767	0,006788	0,145402	0,166649	0,007938	0,024551	0,005803	0,008590
0,012057	0,005974	0,024366	0,009582	0,148386	0,007888	0,159952	0,234120	0,007975	0,023268	0,005762	0,009813
0,011544	0,006127	0,028871	0,009613	0,174072	0,008095	0,149034	0,143590	0,008017	0,022233	0,006818	0,008603
0,027138	0,005736	0,056417	0,009575	0,162385	0,007809	0,145022	0,164188	0,007977	0,022359	0,005725	0,012130
0,011984	0,005482	0,024361	0,009856	0,160033	0,008462	0,156838	0,168470	0,008016	0,023233	0,005629	0,019430
0,013370	0,005227	0,033973	0,010016	0,157685	0,008789	0,145633	0,143129	0,008032	0,023701	0,004874	0,008508
0,011771	0,004972	0,033519	0,010176	0,155338	0,009116	0,144427	0,145140	0,008048	0,022496	0,004859	0,008684
0,011594	0,004718	0,033066	0,010336	0,152995	0,009443	0,143222	0,143086	0,008064	0,024551	0,004973	0,008831
0,012065	0,004463	0,032613	0,010497	0,150643	0,009769	0,152016	0,145377	0,008080	0,022310	0,006077	0,008745
0,012453	0,004209	0,032159	0,010657	0,168295	0,010097	0,160811	0,142749	0,008096	0,022455	0,006110	0,009346
0,011465	0,006954	0,031706	0,010817	0,155948	0,010424	0,139605	0,143177	0,008112	0,022412	0,006143	0,010109
0,015553	0,006699	0,031253	0,010977	0,143654	0,010751	0,138399	0,146615	0,008128	0,024584	0,006175	0,021572
0,012196	0,006445	0,030799	0,011137	0,141253	0,011078	0,137194	0,147851	0,008144	0,014756	0,006208	0,011208
0,012212	0,006199	0,030346	0,011298	0,158905	0,011404	0,135988	0,147664	0,008159	0,014429	0,006241	0,008572
0,012392	0,005935	0,029892	0,011458	0,156558	0,011731	0,144783	0,152178	0,008176	0,014101	0,006274	0,010204
0,012058	0,005681	0,029439	0,011618	0,134213	0,012058	0,143577	0,144192	0,008192	0,022577	0,006306	0,012475
0,012031	0,006426	0,058985	0,011778	0,161863	0,012385	0,132371	0,143248	0,008207	0,014450	0,006339	0,008652
0,014136	0,006172	0,028532	0,011938	0,169515	0,012712	0,151166	0,145342	0,008223	0,019117	0,006372	0,009060
0,012909	0,006917	0,028078	0,012099	0,177168	0,013039	0,139962	0,199771	0,008239	0,018789	0,006404	0,008725
0,012013	0,006662	0,027625	0,012259	0,154824	0,013366	0,138755	0,143600	0,008255	0,018461	0,006437	0,008839
0,012560	0,006234	0,025518	0,009680	0,174477	0,008808	0,146369	0,144802	0,010590	0,010967	0,006766	0,008752
0,012505	0,006358	0,025547	0,009510	0,143430	0,010348	0,148319	0,142800	0,007868	0,010988	0,004751	0,008668
0,012419	0,006208	0,025186	0,009600	0,149218	0,008788	0,146045	0,145352	0,019340	0,019111	0,004746	0,020269
0,012510	0,006240	0,025300	0,009535	0,142630	0,008900	0,146152	0,161566	0,007873	0,010853	0,004757	0,008864
0,012640	0,006265	0,043000	0,009484	0,142653	0,008816	0,145985	0,165222	0,007896	0,012115	0,004797	0,008658

0,012388	0,006147	0,025134	0,020539	0,143425	0,008971	0,146164	0,143974	0,011329	0,010886	0,004709	0,008839
0,025730	0,014680	0,024900	0,022493	0,145600	0,021144	0,146221	0,151380	0,007900	0,010940	0,004899	0,008627
0,012177	0,006141	0,048233	0,009600	0,144500	0,008828	0,314940	0,145789	0,007928	0,010960	0,005015	0,008790
0,012525	0,006142	0,025059	0,019410	0,160535	0,008726	0,146120	0,162233	0,007935	0,011310	0,004844	0,016430
0,012688	0,007337	0,054354	0,009600	0,143421	0,008766	0,146760	0,143063	0,007887	0,026170	0,004690	0,013200
0,013454	0,007182	0,024870	0,023078	0,146770	0,008952	0,146090	0,142937	0,007836	0,012440	0,004725	0,009181
0,012856	0,006190	0,025175	0,009487	0,175157	0,008822	0,148400	0,145558	0,019000	0,011010	0,004774	0,009504
0,012383	0,006288	0,054683	0,021640	0,146514	0,010204	0,146536	0,144680	0,007896	0,010871	0,004780	0,010551
0,027470	0,006100	0,025033	0,009498	0,146344	0,021081	0,146750	0,155020	0,019030	0,019670	0,004734	0,008589
0,011706	0,006296	0,024973	0,009522	0,146240	0,008795	0,158730	0,159900	0,009941	0,022250	0,004805	0,008591
0,011746	0,006180	0,025183	0,009514	0,146872	0,008810	0,146026	0,144790	0,008060	0,010983	0,004703	0,008549
0,011763	0,006237	0,024852	0,009653	0,302065	0,008801	0,146251	0,160690	0,007866	0,011124	0,004736	0,008890
0,011566	0,006204	0,025000	0,009497	0,151886	0,008849	0,146098	0,197083	0,019111	0,025890	0,011336	0,009985
0,011666	0,006172	0,025194	0,009573	0,162623	0,008851	0,339900	0,143647	0,007895	0,010912	0,004763	0,008490
0,015566	0,006309	0,024940	0,009600	0,145877	0,008824	0,145920	0,215449	0,007990	0,011206	0,004718	0,008563

Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Test 11	Test 12
0,012420	0,003182	0,025033	0,001915	0,013469	0,004733	0,013286	0,144060	0,000373	0,011193	0,002239	0,008721
0,011824	0,003202	0,025234	0,001782	0,013300	0,002126	0,013349	0,143500	0,000398	0,011099	0,002270	0,009021
0,011995	0,007076	0,024927	0,004051	0,013382	0,002151	0,013500	0,178659	0,000361	0,011054	0,002224	0,009149
0,011966	0,007189	0,045010	0,001804	0,013072	0,002126	0,013329	0,143368	0,000379	0,011472	0,002181	0,008501
0,011857	0,003125	0,043980	0,001849	0,013214	0,004900	0,014228	0,143893	0,000372	0,016978	0,004817	0,008546
0,011896	0,003140	0,025012	0,001811	0,013429	0,004893	0,013275	0,178483	0,000421	0,011175	0,002216	0,009046
0,011856	0,007088	0,024935	0,001832	0,013203	0,002112	0,013620	0,158180	0,000383	0,011168	0,003072	0,008796
0,014988	0,003150	0,026024	0,001816	0,013388	0,002269	0,014260	0,148500	0,000395	0,011119	0,002271	0,008449
0,015307	0,004882	0,026057	0,001814	0,013310	0,002128	0,013362	0,169530	0,000380	0,025626	0,002192	0,010027
0,014744	0,003185	0,045028	0,001681	0,018361	0,002129	0,031000	0,146000	0,000394	0,011773	0,002190	0,008726

0,015718	0,003126	0,052429	0,001602	0,013890	0,002188	0,013080	0,191562	0,000395	0,011000	0,002545	0,008599
0,029505	0,003198	0,043652	0,001523	0,013194	0,002759	0,013518	0,148306	0,000397	0,011197	0,002540	0,010743
0,028241	0,003149	0,025320	0,001444	0,013366	0,002201	0,017101	0,146699	0,000399	0,011097	0,002536	0,008538
0,015057	0,003194	0,027688	0,001755	0,013844	0,002153	0,013315	0,161621	0,000401	0,011179	0,002532	0,008481
0,036165	0,003238	0,025765	0,001825	0,013278	0,002094	0,013336	0,162284	0,000403	0,012900	0,002528	0,009119
0,015376	0,003314	0,025239	0,001727	0,013381	0,002175	0,013540	0,162947	0,000404	0,012933	0,002524	0,009142
0,014844	0,003135	0,026263	0,001843	0,013521	0,013786	0,013180	0,163610	0,000406	0,012967	0,002520	0,009164
0,014796	0,007378	0,024860	0,001755	0,013978	0,002156	0,013341	0,164273	0,000408	0,013000	0,002516	0,009186
0,015667	0,004154	0,033764	0,001811	0,013520	0,002149	0,013173	0,164936	0,000410	0,013033	0,002512	0,009208
0,016077	0,007628	0,046900	0,001536	0,013889	0,002566	0,017244	0,165599	0,000412	0,013066	0,002507	0,009230
0,014878	0,003250	0,033674	0,001502	0,013909	0,002246	0,015140	0,143368	0,000414	0,013099	0,002503	0,009253
0,014715	0,003304	0,033819	0,001468	0,013929	0,002102	0,015182	0,143893	0,000415	0,013132	0,002499	0,009275
0,014673	0,003381	0,033964	0,001434	0,013949	0,002122	0,015223	0,178483	0,000417	0,013165	0,002495	0,009297
0,029366	0,003167	0,037108	0,001400	0,013969	0,002128	0,015265	0,158180	0,000419	0,013198	0,002491	0,009319
0,015465	0,003283	0,035253	0,001866	0,013989	0,002875	0,015307	0,148500	0,000421	0,013231	0,002487	0,009341
0,014954	0,003736	0,034397	0,001832	0,014009	0,002864	0,015349	0,169530	0,000423	0,013264	0,002483	0,009364
0,015348	0,003158	0,039542	0,001898	0,014029	0,002853	0,015390	0,146000	0,000424	0,013298	0,002478	0,009386
0,014948	0,003347	0,034687	0,001764	0,014049	0,002841	0,015432	0,191562	0,000426	0,013331	0,002474	0,009408
0,015852	0,003271	0,034831	0,001505	0,014068	0,002830	0,015474	0,171566	0,000428	0,013364	0,002470	0,009430
0,016578	0,003814	0,034976	0,001485	0,014088	0,002819	0,015515	0,172229	0,000430	0,013397	0,002466	0,009452
0,016726	0,003200	0,035120	0,001465	0,014108	0,002808	0,015557	0,172892	0,000432	0,013430	0,002462	0,009475
0,015498	0,003174	0,026024	0,001445	0,014128	0,002796	0,015599	0,173555	0,000433	0,013463	0,002458	0,009497
0,015696	0,003146	0,026057	0,001425	0,014148	0,002785	0,015641	0,143368	0,000435	0,011496	0,002454	0,009519
0,015764	0,003139	0,046028	0,001404	0,014168	0,002774	0,015682	0,143893	0,000437	0,011529	0,002450	0,009541
0,015424	0,004457	0,053429	0,001884	0,014188	0,002762	0,015724	0,178483	0,000439	0,011562	0,002445	0,009563
0,023280	0,003178	0,043652	0,001864	0,014208	0,002751	0,015766	0,158180	0,000441	0,011595	0,002441	0,009586
0,023060	0,003203	0,025320	0,001844	0,014228	0,002740	0,015807	0,148500	0,000443	0,011629	0,002437	0,008608
0,014908	0,007217	0,027688	0,001724	0,014248	0,002729	0,015849	0,169530	0,000444	0,011662	0,002433	0,008630

0,014869	0,003158	0,025765	0,001603	0,014268	0,002717	0,015891	0,146000	0,000446	0,011695	0,002429	0,008652
0,014569	0,003340	0,025239	0,001683	0,014288	0,002706	0,015933	0,191562	0,000448	0,011728	0,002425	0,008674