Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Optimalizace databází v internetovém obchodě

Jiří Labuta DiS.

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jiří Labuta, DiS.

Informatika

Název práce

Optimalizace databází v internetovém obchodě

Název anglicky

Optimizing databases in Internet sales

Cíle práce

Bakalářská práce je zaměřena na problematiku optimalizací databázových evidencí v oblasti internetového obchodování. Cílem práce bude:
a) objasnit teoretické principy relačně databázové technologie se zřetelem na použití v internetovém obchodování,
b) zmapovat současnou úroveň této problematiky a vymezit požadavky na ni kladené,
c) navrhnout řešení těchto vymezených požadavků,
d) navržené řešení ověřit v rámci funkčního prototypu,
e) ověřené záležitosti zobecnit pro další možná použití.

Metodika

Použitá metodika zadané bakalářské práce bude založena na studiu a analýze dostupných informačních zdrojů a existujících řešení v dané oblasti. Stěžejní pro vypracování této závěrečné práce budou metody a techniky relačně databázové technologie v kontextu s optimalizací internetových databází. Navrhované řešení bude zohledňovat identifikované požadavky a očekávání spojená s řešenou záležitostí. Na podkladě syntézy teoretických poznatků a dosažených výsledků budou formulovány závěry této bakalářské práce a následně zobecněny pro další možná použití.

Doporučený rozsah práce

40-50 stran

Klíčová slova

relačně databázová technologie, optimalizace databázové evidence, Internový obchod, SQL, MySQL

Doporučené zdroje informací

BEGG, C., CONOLLY, T., HOLOWCZAK, R.: Mistrovství databáze, profesionální průvodce tvorbou efektivních databází. Computer Press. Brno 2009. ISSBN 978-80-251-2328-7

BRYLA, B., LONEY, K.: Mistrovství v Oracle Database 10g. Computer Press Brno 2006. EAN 978802512779

HERMANDEZ, M.: Návrh databází, GRADA 2005. ISBN 80-247-0900-7.

LACKO, L: ORACLE. Správa, programování a použití databázového systému. Computer Press Brno 2007. EAN 97880251149002.

LONEY, K.: Oracle Database, kompletní průvodce. Computer press Brno 2010. ISBN 978-80-251-2489-5

MOLINARO, A.: SQL Kuchařka programátora. Computer Press. Brno 2009. ISBN 978-80-251-2617-2

POKORNÝ, J.: Databázové systémy. ČVUT Praha 2013. ISBN 978-80-01-05212-9

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

doc. Dr. Ing. Václav Vostrovský, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 13. 03. 2016

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Optimalizace databází v internetovém obchodě" jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 6. 3. 2016 _____

Poděkování

Rád bych touto cestou poděkoval doc. Dr. Ing. Václavovi Vostrovskému, Ph.D. za jeho konzultace a vedení mé práce. Rovněž děkuji také rodině, která mi poskytla podporu při psaní této práce.

Optimalizace databází v internetovém obchodě

Optimizing databases in Internet sales

Souhrn:

Bakalářská práce řeší jedno z velmi aktuálních témat současné doby. Optimalizace v internetovém obchodě je jednou ze základních vlastností úspěšných a prosperujících internetových obchodů. Vzhledem k možnostem využití relačních databázových technologií se práce zaměřuje na využití databázového systému MySQL, která v této oblasti patří mezi nejpoužívanější.

První část práce poukazuje na vzrůstající trend internetového obchodování ve světě i v České republice do současné doby s nastíněním směru, kterým se internetové obchodování ubírá.

Další část práce popisuje teoretické principy, které ovlivňují výkon databázového systému MySQL a možnosti sledování výkonu. Rovněž práce vystihuje nejdůležitější techniky optimalizace.

V praktické části jsou popsány jednotlivé optimalizační techniky, které jsou aplikovány dle teoretických principů na středně velkých internetových obchodech s praktickými zkušenostmi autora. Tyto principy jsou aplikovány na produktivních internetových obchodech.

Klíčová slova:

Databáze, databázová evidence, optimalizace databázových evidencí, normalizace, internetový obchod, MySQL

Summary:

The bachelor´s thesis is focused on one of the current topics of the present day. Optimization in online shopping is one of the fundamental features of successful and prosperous Internet shops. Considering the possibilities to use the relational database technologies the

paper concentrates on the MySQL database system which belongs among the most used in this area.

The first part of the thesis looks at the growing trend of Internet trading worldwide as well as in the Czech Republic and outlines the direction online shopping is taking. The second part of the paper describes theoretic principles influencing the performance of the MySQL database system and the possibilities to track its performance. There are also demonstrated the most important techniques of optimization.

The practical part illustrates individual optimization techniques that are applied to middle-sized Internet shops according to theoretic principles and include the author´s practical experience. These principles are applied to productive Internet shops.

Keywords:

Database, database evidence, optimizing of database records, normalization, internet sales, MySQL

Obsah

Seznam obrázků

1 Úvod

V současné době je člověk téměř na každém kroku obklopen výpočetní technikou, která se stává nedílnou součástí našich životů. Jedná se o pracovní stanice, notebooky, tablety, chytré telefony a jiné zařízení, bez kterých si dnes neumíme představit běžný den. Zároveň ceny mobilních připojení těchto zařízení k internetu jsou nastaveny od prodejců, tak, že si je může většina dovolit a tak není divu, že jsou velmi oblíbené on-line aplikace.

Velké popularitě se samozřejmě těší online obchodování. Lidé si mohou nakoupit téměř cokoliv z tepla domova, po cestě do práce apod. Rovněž mohou lépe porovnat nabízené produkty mezi jednotlivými obchody a tím získat úsporu nejen časovou, ale hlavně finanční.

Tato práce je zaměřena na malé až středně velké internetové obchody, jejichž roční obrat nepřesahuje hranici 10 miliónů korun. V této kategorii provozovatelé většinou využívají produktů s volnou či open-source licencí, aby jim při nákupu internetového obchodu nevznikl další náklad. Navíc v této kategorii je nejčastější používaným databázovým systémem MySQL, na který je tato práce zaměřena.

Aby se uživatelé na daný internetový obchod vraceli, je třeba zajistit jeho rychlou funkčnost a odezvu. Obvykle dochází k porovnání několika obchodů zároveň a velmi pomalý obchod nemá v dnešní době šanci uspět.

Tyto internetové obchody jsou ve velké míře založeny na již vytvořených redakčních systémech, jejichž univerzálnost usnadňuje spuštění, ale přizpůsobení aplikace pak bývá řešeno profesionály. Na druhou stranu zde existuje i silné zastoupení aplikací tvořených na míru, ale vzhledem k obratu to nejsou aplikace týmu programátorů, ale spíše programátorů - jedinců.

Úvodem je třeba přesně vymezit pojem optimalizace. Ačkoliv existuje mnoho definic, tato práce se bude zabývat pouze optimalizací z hlediska rychlosti databáze. V oblasti internetových obchodů, které jsou v práci zmíněny, hraje rychlost významnou roli. Rychlost je také velmi častý argument společností, které nabízí služby zrychlení stránek, SEO apod. Přesto v této oblasti je rychlost zásadní konkurenční výhodou, neboť nabídka internetových obchodů je tak rozmanitá, že v této kategorii nalezneme jen minimum obchodů se speciálním zbožím, či velmi drahým zbožím, kde zákazníka neodradí pomalé načítání.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem bakalářské práce zaměřené na optimalizaci databází v internetových obchodech je objasnit teoretické principy relačně databázové technologie, zmapovat současnou úroveň problematiky internetového obchodování, vymezit požadavky kladené na aplikace internetových obchodů, navrhnout řešení těchto požadavků, navržené řešení ověřit a ověřené záležitosti zobecnit pro další použití.

V rámci takto stanovených cílů bude dílčím cílem zmapovat současnou úroveň internetových obchodů s předpokládaným vývojem v oblastech prodeje i konkurenceschopnosti v daných oblastech. Vzhledem rozsáhlosti a rozmanitosti internetových obchodů se zaměřit na nejširší pásmo těchto aplikací a zmapovat důvody potřebnosti optimalizace. Rovněž vymezit požadavky kladené na internetové obchody ze strany uživatelů i poskytovatelů.

Objasnit teoretické principy databází pro internetové obchody ve spojení s relačně databázovou technologií a zaměřením se na databázový systém MySQL.

Na základě vymezení požadavků kladených provozovateli a uživateli aplikace navrhnout na základě teoretických poznatků metody a možnosti optimalizace databázového systému MySQL za účelem zrychlení a zkvalitnění aplikace.

Popsané teoretické principy aplikovat na praktickém příkladu internetového obchodu. Veškeré získané závěry zobecnit pro další možná využití v praxi.

2.2 Metodika

Práce byla vypracována na základě metod relační datové technologie, datové integrity a datové normalizace.

Pro vypracování této bakalářské práce v kontextu ke stanoveným cílům byla zvolena a použita následující metodika.

1) Shromáždění informačních zdrojů existujících v oblasti relační databáze a zmapování dílčích teoretických principů v oblasti optimalizace internetových obchodů.

2) Porovnání funkčních i méně funkčních řešení včetně kontaktu s provozovateli a zákazníky daných aplikací.

3) Zhodnocení očekávání funkčnosti internetových obchodů ze strany provozovatelů a zákazníků

4) Navržení řešení na modelovém příkladu zohledňujícím požadavky kladené na aplikaci a odstranění identifikovaných problémů.

5) Zhodnocení výsledků provedených úprav ve spojení s očekáváním v dané oblasti.

6) Zobecnění závěrů pro další možné využití v oblastech optimalizace.

3 Teoretická východiska

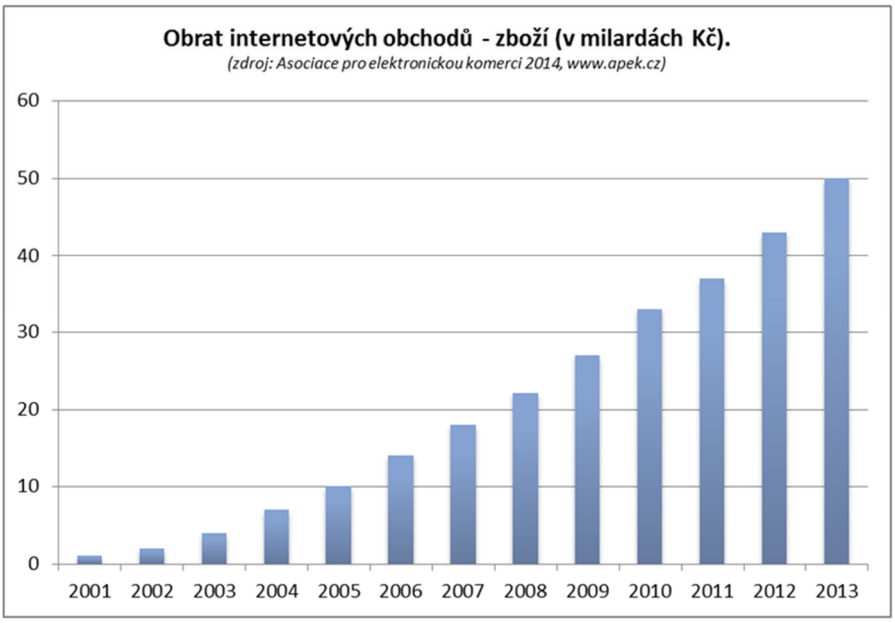
3.1 Internetový obchod

Internetový obchod neboli e-obchod, či e-shop je webová aplikace, která představuje jistou formu nákupu či prodeje zboží nebo služeb elektronickou cestou. Elektronickou cestou je zde myšlen internet. Internetový obchod slouží k nabídce zboží, kde má zákazník možnost objednat si dané zboží (nebo službu). Zároveň e-obchod zprostředkovává data, eviduje platby a poskytuje další informace o výrobcích. Většinou se jedná o sadu skriptů, která pracuje s databází, ve které jsou evidovány detaily o zboží a službách. Funkcí je, aby co nejvíce ulehčili administrátorovi obchodu práci při evidenci dat o zboží, zákaznících, platbách, skladové evidenci apod. [4]

3.1.1 Historie internetových obchodů

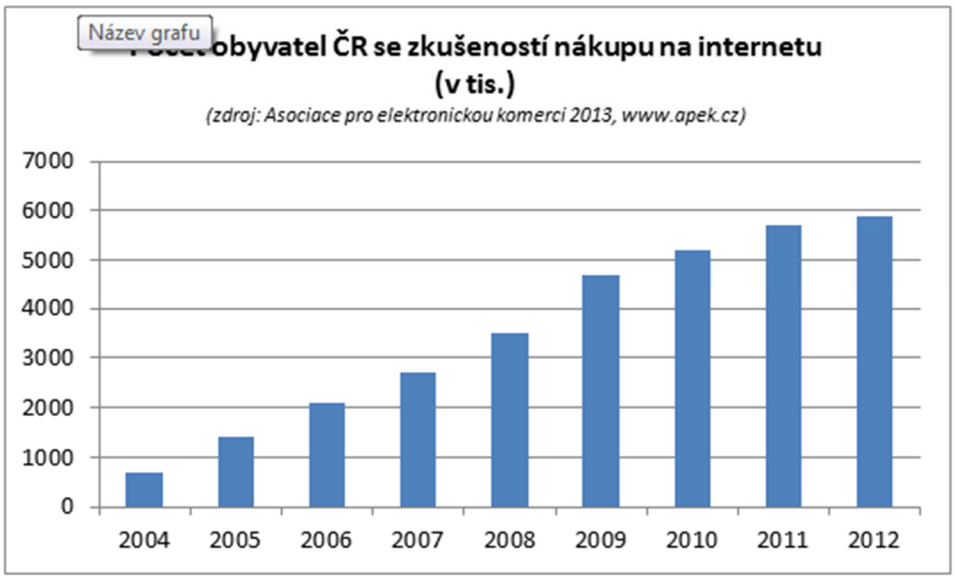
Historie internetových obchodů sahá do roku 1992, kdy se v USA první on-line prodejní komoditou stal prodej CD. Po těchto následovaly dárkové předměty a knihy. Ostatní komodity jako například elektronika následovaly až později. Internetové obchody, tak jak je známe dnes, začaly vznikat v letech 1994 až 1995 a za jejich rozvoj může v podstatě boom protokolů http a www. [2]

V České republice se první internetové obchody začaly objevovat v roce 1996. Od této doby internetové obchodování v České republice stále roste. Alespoň jednou do roka nakoupí na internetu 79% občanů. Samozřejmě, že největším důvodem je úspora času. To také je důvod pro 78% občanů, čili je vidět, že to je téměř shodné s procentem občanů, kteří nakupují na internetu. Aktuálně je na českém trhu více než 30 tisíc e-shopů.



Obrázek 1 - Obrat internetových obchodů v ČR - zboží (v miliardách Kč)

(Zdroj: apek.cz (3))



Obrázek 2 - Počet obyvatel v ČR se zkušeností nákupu na internetu (v tis.)

(Zdroj : apek.cz (3))

V podstatě každý druhý e-shop je vytvořen na komerčním řešení a cca. 40% obchodů stojí na open-source. Jen necelá desetina vznikla individuálně. Přesto drtivá většina internetových obchodů pouze přežívá, a proto také životnost není příliš dlouhá.

Pokud se ještě zaměříme na výhody využívání internetových obchodů, kterými jsou úspora času, pohodlí a hlavně velmi rychlé porovnání produktů a cen jednotlivých obchodů.

Rovněž je neméně významným faktorem i nižší cena a širší výběr [3]. Internetové obchodování mimo výhod doprovází také některé nevýhody, či spíše obavy zákazníků. Tou největší obavou pro většinu Čechů je zneužití platební karty. Jednoznačnou nevýhodou je nemožnost vyzkoušet, či „osahat“ zboží, což platí hlavně pro oblečení. Samozřejmě i rada prodavače může zákazníkovi při nákupu pomoci vybrat ten správný produkt, nebo osvětlit jeho použití. Na toto však většina internetových obchodů reaguje nejrůznějšími recenzemi prodejců, zákazníků či výrobců, nebo vytvořenými fóry mezi zákazníky. Přesto rozsah výhod, které internetové obchody mají, jim napovídá, že ani v budoucnu se nedá předpokládat snížení nakupování přes internet, ba naopak.

3.1.2 Varianty dodávaných internetových obchodů

V případě pořízení internetového obchodu je několik možností, kterých lze využít. Finančně nejnáročnější, ale z hlediska pořizovatele nejlépe přizpůsobenou verzí je naprogramování internetového obchodu tzv. na míru. Mimo finanční náročnosti je třeba u této verze nutné počítat s tím, že každý doplněk, na který se při počátečním programování zapomnělo se musí opět doprogramovat a tak je třeba počítat s částkou v řádech desítek tisíc korun.

Další variantou je koupě „krabicového“ systému. Ten bývá nejčastěji modulární, tzn. že pořizovatel si může zakoupit pouze ty části, které hodlá využívat. Tento systém je podstatně levnější než vlastní programování a i dokoupení dalších modulů není finančně tak náročné. Výhodou je i rychlost nasazení. Ve většině případů to softwarové firmy nabízí včetně hostingu. Pak se nemusí ani jednat o jednorázovou platbu, ale např. měsíční platby. Zde však pořizovatel nemá přístup ke zdrojovým souborům. Výhodou modulárních systému je bezpochyby také velké zobecnění a tak podobný systém nabízí velké množství možných funkcí.

Poslední variantou jsou Open source řešení, která jsou volně ke stažení, nebo lze využít systémy pod doménou poskytovatele, která však ve využívaných internetových obchodech tvoří mizivou část.

3.2 Databáze MySQL

3.2.1 Základní pojmy

Databáze – Databázi lze chápat jako úložiště údajů, které jsou zpracovávány nezávisle na aplikačních programech. Data jsou do databáze ukládána v jednotlivých záznamech (řádcích).

Databázový systém – Databázový systém zahrnuje údaje, které jsou uloženy a spravovány v databázi, a také software pro přístup k těmto údajům. Databázový systém je tvořen jednak systémem řízení báze dat (SŘBD, nebo anglicky Database Management System, DBMS) a databází. Databázové systémy mohou být:

- **hierarchické a síťové** - u těch jsou aplikační programy závislé na databázi, z čehož vyplývá například problematická údržba
- **relační** - pro ně je typická neprocedurální manipulace s daty, ukládání dat s pevnou strukturou v tabulkové formě
- **objektové** - ty používají složité datové struktury a pravidla založená na obchodní logice

SQL jazyk – strukturovaný dotazovací jazyk (Structured Query Language) používaný v relačních databázích pro práci s daty.

Součásti jazyka SQL - Jazyk SQL má čtyři hlavní části. Jsou to:

- **Data Definition Language (DDL)** - jazyk pro definici dat sloužící k vytváření a odstraňování databází, k přidávání a odstraňování tabulek, k vytváření indexů a k modifikaci všech těchto objektů
- **Data Manipulation Language (DML)** - jazyk pro manipulaci s daty, což je množina příkazů, které spolupracují s databázovým systémem a s vlastní databází. Příkazy jazyka DML zajišťují vyhledávání, vkládání a odstraňování dat
- **Správa transakcí** - příkazy, s jejichž pomocí lze zpracovávat určitý soubor příkazů jako jednu ucelenou jednotku (transakci)

Pokročilé funkce - slouží pro vložení jazyka SQL do obecných programovacích jazyků, pro definování speciálních pohledů na podkladová data a pro přidělování a odebrání práv přístupu k databázovému systému a jednotlivým databázím. Rovněž jsou zde obsaženy funkce k zajištění integrity systému. [5]

3.2.2 Historie MySQL

Prvopočátky MySQL lze najít již v roce 1979, kdy tvůrce Ulf Michael Widenius (zvaný Monty) pracoval pro švédskou IT společnost TcX. V této době vytvořil UNIREG, což je terminálové rozhraní využívající low-level spojení ISAM.

V roce 1994 společnost TcX začala vyvíjet webové aplikace. Pro tyto však UNIREG nebyl vhodný a tak za pomoci Davida Hughese vznikl mSQL. Bohužel první verze mSQL postrádala indexování, které pro správu velkých datových úložišť byly zcela zásadní. Následně se Widenius a TcX rozhodlo vyjít z podstatné práce na UNIREG vytvořením nové API podobné v mSQL, avšak za použití nové koncepce v roce 1995 vznikl produkt s přezdívkou MySQL. MySQL by vypuštěn zdarma jako open-source a další finanční prostředky firma získala prodejem podpory pro firmy, které chtěli MySQL využít. Zároveň společnost TcX změnila název na MySQL AB. Během dalších let se MySQL stal produktem světové úrovně a byl světově nejpoužívanější opensource databázovým systémem. [6]

Sun Microsystems v roce 2008 koupila za jednu miliardu dolarů společnost MySQL AB. V roce 2009 pak odkoupila společnost Oracle Sun Microsystems za cenu 7,4 miliard dolarů a do současnosti je vývojářem MySQL právě společnost Oracle.

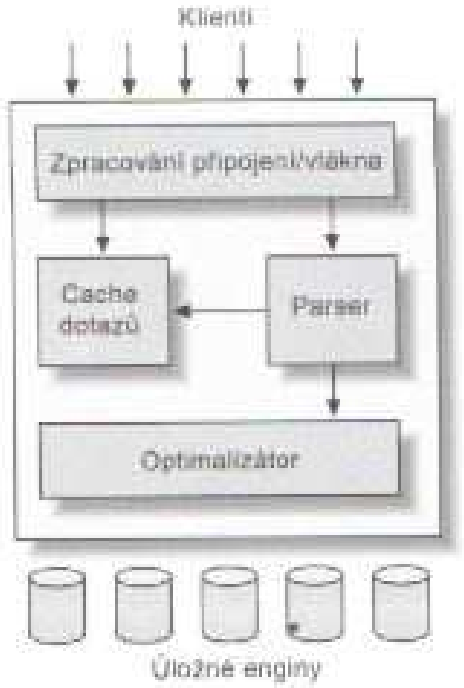
3.2.3 Popis databáze MySQL

MySQL je open-source SQL relační databázový management systém (RDBMS), který lze využít zdarma [6]. Architektura MySQL je velmi odlišná od jiných databázových serverů. Má velmi široký záběr a je užitečná pro řešení mnoha různorodých úloh. Přestože nemůžeme řadit MySQL mezi ty nejkvalitnější, tak přesto je dostatečně flexibilní a to i pro práci ve velmi náročných prostředích. Zároveň může MySQL posílit vkládané aplikace, datové sklady, software pro indexování obsahu, redundantní systémy s vysokou dostupností atd. [7]

Architektura MySQL serveru je složena ze tří vrstev. První vrstvou jsou služby, které obsluhují většinu potřebných nástrojů klient/server, a které jsou založeny na síti. Jedná se například o zpracování připojení, autentizaci, bezpečnost atd.

V druhé vrstvě je většina operací, které fungují jako tzv. mozek MySQL. Jedná se o kód pro rozbor dotazu, analýzu, optimalizaci a veškeré zabudované funkce (např. datum a čas). V této vrstvě se nachází veškerá funkcionalita, která je poskytována prostřednictvím úložných enginů.

Třetí vrstva obsahuje úložné enginy. Ty ukládají a získávají všechna data uložená v MySQL. 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. Úložné enginy až na InnoDB nedělají rozbor SQL a nekomunikují mezi sebou. V podstatě pouze odpovídají na dotazy od serveru. [7]



Obrázek 3 - Architektura MySQL

(Zdroj: Optimalizace pro vysoký výkon (7))

3.3 Funkce ovlivňující výkon MySQL

S každou verzí MySQL, která je vydána se výkon zvyšuje. Jen u nejnovější verze 5.7 je vydavatelem garantován až 3x vyšší výkon. Ve své práci se však budu držet nižších verzí 5.X, neboť tyto verze jsou implementovány v internetových obchodech, které jsou v práci zmíněny.

Rovněž i vzhledem k velkému rozvoji internetových obchodů využívajících MySQL je třeba řešit také optimalizaci. Důvody jsou více než zřejmé. Rychlý internetový obchod je jistou konkurenční výhodou. Zákazník vyžaduje, aby veškerá požadovaná data získal velmi rychle.

3.3.1 Možnosti optimalizace na straně serveru

Konfigurace MySQL serveru je nastavena, že nevyužívá přílišné množství hardwarových zdrojů. Základní konfigurace v podstatě nepotřebuje více zdrojů než je třeba pro spuštění MySQL a pak také pro spuštění jednotlivých dotazů.

Velmi důležité je zajištění, aby MySQL využíval správně paměť. Vždy je velmi účelné nastavit horní velikost paměti, kterou může MySQL využít. Vždy existuje limit ve velikosti paměti, který může být na daném systému pro MySQL použitelný. Je to samozřejmě přímo závislé na velikosti fyzicky dostupné paměti pro daný server. Zároveň je třeba akceptovat limity operačního systému, které mají své restrikce. MySQL běží jako vícevláknový proces, což je hlavně u 32bitových operačních systémů omezující. Například u 32 bitového jádra Linuxu

může jediný proces adresovat paměť na hodnoty mezi 2,5 – 2,7 GB. Nejde zde však pouze o limity jednotlivých procesů, ale také jde o velikost zásobníku apod. I na 64 bitových serverech existují limity, jako například velikosti bufferů, avšak každá nová verze MySQL těží z rozvoje výkonu současného hardware.

Pro správné vyhrazení dostatečné velikosti paměti je nutné znát, kolik jí může MySQL spotřebovávat v tzv. špičkách. Paměť potřebná pro udržení vláken (otevřeného připojení) nebývá nijak velká, přesto je třeba právě na to brát zřetel. Vykonávání dotazů ve špičkách při nedostatečné paměti by běželo jen velmi špatně a mohlo by v krajním případě dojít i k selhání. Bránit se tomuto jevu můžeme třeba konfigurací maximálního možného počtu připojení.

Dalším ovlivňujícím faktorem, kterému musíme také vyhradit dostatek paměti je operační systém. Obvykle není třeba pro operační systém vyhradit více než 1, či 2 GB. Více paměti by operační systém mohl vyžadovat snad jen pouze pro zálohování. Dobrým vodítkem, zda má systém dostatek paměti, je že systém aktivně neswapuje virtuální paměť na disk.

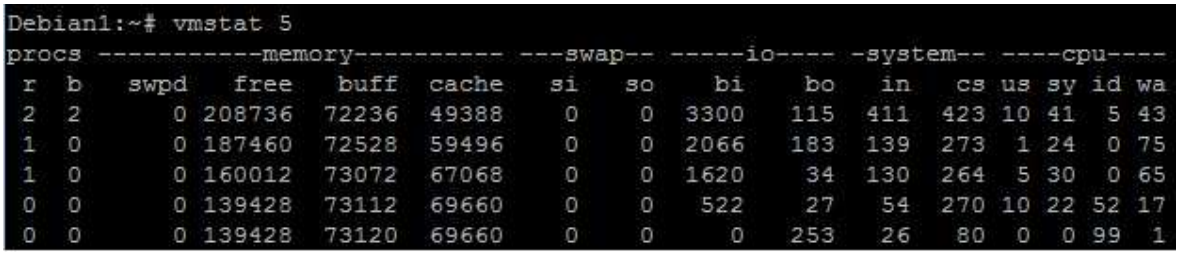
Paměťově nejnáročnější je však potřeba paměti pro různé cache. Pokud je server vyhrazen pouze pro MySQL, pak je pro cache dostupná jakákoliv paměť mimo vyhrazené pro operační systém a zpracování dotazů. Cache je využívána proto, aby se nevyužíval přístup na disk. Cache je řádově rychlejší než přístup k datům umístěným v paměti. Mezi nejdůležitější cache patří:

- Cache operačního systému pro data MyISAM
- Cache klíčů MyISAM
- Buffer pool InnoDB
- Cache dotazů

Výše vyjmenované cache patří pouze mezi ty nejdůležitější, neboť ostatní obvykle nezaberou velké množství paměti.

3.3.2 Možnosti optimalizace na straně operačního systému a hardware

Tak jako v mnoha odvětvích platí, že celý systém pracuje tak dobře, jako jeho nejslabší článek, tak o MySQL lze toto říct také. Velmi často patří právě operační systém a hardware mezi omezující faktory. Nejčastěji limitující součásti pro výkon MySQL jsou CPU, paměť a vstupně-výstupní zařízení. Pro zjištění omezení MySQL serveru lze využít několika nástrojů. Jedná se např. o vmstat, iostat, mpstat.



```
Debian1:~# vmstat 5
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 2  2      0 208736  72236  49388    0    0  3300   115  411  423 10 41  5 43
 1  0      0 187460  72528  59496    0    0  2066   183  139  273  1 24  0 75
 1  0      0 160012  73072  67068    0    0  1620    34  130  264  5 30  0 65
 0  0      0 139428  73112  69660    0    0   522    27   54  270 10 22 52 17
 0  0      0 139428  73120  69660    0    0     0   253   26   80  0  0 99  1
```

Obrázek 4 - Využití vmstat na Linuxu

(Zdroj: http://www.zdrojak.cz/clanky/upgradujeme-mysql-server/ (9))

3.3.2.1 Procesor

Velmi často omezuje výkon nasycenost CPU, ke které dochází v případě, kdy MySQL pracuje s daty, která se vejdou do paměti nebo je možné je načíst z disku tak rychle, jak jsou potřeba. V případě využití obrázku 4 se toto projevuje jednak vysokou hodnotou ve sloupci us (v sekci cpu), což značí celkový čas v procentech strávený vykonáváním uživatelského kódu. Rovněž bude také nenulová hodnota ve sloupci r (v sekci procs), která udává počet procesů čekajících na CPU. Očekávat lze také vysokou hodnotu cs (v sekci system), což je počet přepínání kontextu za sekundu.

Pokud tedy budeme předpokládat, že systém, pro který optimalizujeme procesory, má opravdu zátěž orientovanou na CPU. Obvykle je třeba se rozhodnout mezi rychlostí procesoru či vyššího počtu jader. Současná architektura MySQL má potíže se škálováním na více jader, takže není možné nechat běžet jeden dotaz paralelně přes několik CPU. Výhodnější pak je využití rychlejších CPU na úkor jejich počtu. Lze tedy říci, že nízké latence, neboli rychlé doby odezvy dosáhneme rychlým CPU, protože dotaz používá pouze jediný CPU. Vysokou propustnost v návaznosti na potíže se škálováním opět dosáhneme lépe s menším počtem rychlejších CPU.

Přesto však nastává situace, kdy je výhodnější použití většího množství procesorů. Jde o situaci, kdy na MySQL serveru běží větší množství databází a jestliže k serveru přistupuje větší množství klientů. V tomto případě jsou situace na sobě nezávislé a vyšším počtem CPU tak lze zvýšit propustnost. [7]

3.3.2.2 Paměť

Snížení výkonu celého systému může velmi negativně ovlivnit swapování bloků paměti na disk. Opět to lze vyčíst z výsledku příkazu vmstat zobrazeném na obrázku 4. Pokud jsou sloupce si a so v sekci swap nenulové, je třeba přidat operační paměť. Sloupec si znamená počet bloků za sekundu swapovaných na disk a so počet bloků za sekundu swapovaných z disku. Rozhodně nikdy nechceme, aby stroj výrazně swapoval. Zvýšené množství paměti má pak

výhody i v případě nasycení vstupně –výstupních operací. Je samozřejmě výhodné co nejvíce informací umístit do operační paměti, aby se z pevného disku muselo číst co nejméně. [9]

3.3.2.3 Vstupně-výstupní zařízení

Dalším zásadním způsobem mohou ovlivnit výkon MySQL vstupně-výstupní operace. K nasycení těchto operací dochází, pokud chceme pracovat s velkým množstvím dat, ale ty se nevejdou do operační paměti. Pak je třeba číst z disku. Tento případ lze opět identifikovat např. z vmstat zobrazeného na obrázku 4. Jednak to je vidět ve sloupci b (v sekci procs), který nám ukazuje počet procesů v režimu spánku (čekají na I/O, síť, vstup uživatele), ale také ve sloupci wa (v sekci cpu), což nám značí celkový čas v procentech strávený čekáním na I/O.

Databázové servery používají dva typy diskových vstupně-výstupních operací. Jsou to sekvenční I/O operace a nesekvenční I/O operace. Sekvenční operace jsou mnohem rychlejší než nesekvenční a to bez ohledu na to, zda máme na mysli paměť, či disk.

3.3.3 Možnosti optimalizace na straně aplikace

V mnohých případech může být pro výkon výhodnější, pokud by se prováděly všechny operace mimo MySQL. Problémem, který však může nastat, je že aplikaci trvá příliš dlouho odpovědět na požadavky.

Správci a návrháři si velmi často zjednodušují vývoj použitím volně dostupných systémů, nebo populárních frameworků. Ačkoliv ve většině případů je podstatně snadnější a rychlejší použití cizího systému, který si správce z jakéhokoliv důvodu nevybudoval sám, vystavuje se velkému riziku, že v podstatě neví, co se děje na pozadí aplikace. Níže jsou nejčastější rizika:

- Využití zdrojů. Obvykle není jasné, co využívá CPU, disk, síť, paměť. Zda čísla, které dokáže zjistit jsou, či nejsou rozumná. Velmi často dochází při chybné konfiguraci k nadměrnému spotřebování paměti procesy
- Využití dat. Často se získá více řádků, než je zobrazeno a není známo, co se děje s ostatními. Zda jsou zahozeny, či uloženy do cache pro pozdější využití.
- Využití vhodných nástrojů. Častým problémem může být chybné rozložení práce pro jednotlivé nástroje. Častým problémem by mohlo být, vytváření složitých operací či manipulací s řetězci v databázi. Ta by měla sloužit k počítání řádků a ne k práci třeba s regulárními výrazy. Na to je vhodné použít jiné aplikace.

- Vykonávání mnoha dotazů. Dotazové rozhraní ORM (object-relational mapping) má za následek, že programátoři se nemusí psát úplně s SQL jazykem, čímž vznikají vnořené cykly, místo dotazů obsahují spojení tabulek.

- Zbytečné připojení MySQL. V některých případech je pravděpodobné, že dochází ke zbytečnému připojení aplikace k MySQL, neboť data jsou uložena v cache.

- Vykonávání zbytečných dotazů. Příkladem by mohl být výběr požadované databáze před každým dotazem

- Trvalé připojení k MySQL. Obecně platí, že není dobré využívat trvalé připojení, což by mohlo vést na mnoho připojení. Výjimkou by mohla však být situace špatné kvality sítě (pomalé připojení). [8]

3.4 Nástroje a programy pro sledování výkonu MySQL

Ne vždy vývojáři dodržují pravidla při zakládání databází. Proto jsou později velmi často využívány nástroje, které pomáhají při analýze chyb, nízkém výkonu, ale například i pro testování. MySQL nabízí celou řadu takových nástrojů. Historicky byly využívány nástroje pro příkazový řádek. Tyto jsou stále k dispozici, avšak častěji využívanými jsou spíše grafické nástroje, které jsou pro správce přívětivější.

3.4.1 Nástroje pro příkazový řádek

- mysqladmin – tento příkaz poskytuje funkce pro správu a diagnostiku databázového serveru. Lze využít pro kontrolu konfigurace, ale také pro zjištění aktuálního stavu na serveru, mazání či vytváření databáze apod. [10] Mezi příkazy, které mysqladmin nabízí patří např. create (vytvoření databáze), debug (zápis informací do protokolu při ladění), extendedstatus (zobrazení stavu proměnných a jejich hodnot) atd.

- SHOW – těchto příkazů je hned několik a dokáží konkrétně hlásit informace o výkonu. Jsou to např.:

    - SHOW ENGINES (zobrazuje informace o stavu úložišť na serveru). Je to dobré pro kontrolu správného typu úložného enginu.

Engine	Support	Comment	Transactions	XA	Savepoints
FEDERATED	NO	Federated MySQL storage engine	*NULL*	*NULL*	*NULL*
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine (anything you write to it...	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary...	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO
InnoDB	DEFAULT	Supports transactions, row-level locking, and fore...	YES	YES	YES
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO

Obrázek 5 - Výsledek dotazu SHOW ENGINES

(Zdroj: Autor)

- SHOW VARIABLES (zobrazuje hodnoty systémových proměnných).

Variable_name	Value
auto_increment_increment	1
auto_increment_offset	1
autocommit	ON
automatic_sp_privileges	ON
back_log	50
basedir	E:/Xampp/mysql
big_tables	OFF
binlog_cache_size	32768
binlog_direct_non_transactional_updates	OFF
binlog_format	STATEMENT
binlog_stmt_cache_size	32768
bulk_insert_buffer_size	8388608
character_set_client	utf8
character_set_connection	utf8
character_set_database	latin1
character_set_filesystem	binary
character_set_results	utf8
character_set_server	latin1
character_set_system	utf8
character_sets_dir	E:\Xampp\mysql\share\charsets\
collation_connection	utf8_general_ci
collation_database	latin1_swedish_ci
collation_server	latin1_swedish_ci
completion_type	NO_CHAIN
concurrent_insert	AUTO
connect_timeout	10
datadir	E:\Xampp\mysql\data\
date_format	%Y-%m-%d
datetime_format	%Y-%m-%d %H:%i:%s
default_storage_engine	InnoDB

Obrázek 6 - Výsledek dotazu SHOW VARIABLES

(Zdroj: Autor)

- SHOW STATUS (zobrazuje statistické údaje o stavu serveru v reálném čase)

Variable_name	Value
Aborted_clients	0
Aborted_connects	2
Binlog_cache_disk_use	0
Binlog_cache_use	0
Binlog_stmt_cache_disk_use	0
Binlog_stmt_cache_use	0
Bytes_received	183
Bytes_sent	133
Com_admin_commands	0
Com_assign_to_keycache	0
Com_alter_db	0
Com_alter_db_upgrade	0
Com_alter_event	0
Com_alter_function	0
Com_alter_procedure	0
Com_alter_server	0
Com_alter_table	0
Com_alter_tablespace	0
Com_analyze	0
Com_begin	0
Com_binlog	0

Obrázek 7 - Výsledek dotazu SHOW STATUS

(Zdroj: Autor)

- Dalšími příkazy pak jsou SHOW INDEX (zobrazuje informace o indexech). Strategie indexování je pro výkon velmi důležitá a níže se jí bude práce ještě zabývat.
- SHOW INNODB STATUS (Poskytuje informace a hlášení o stavu). Od verze 5.5. je odstraněn.

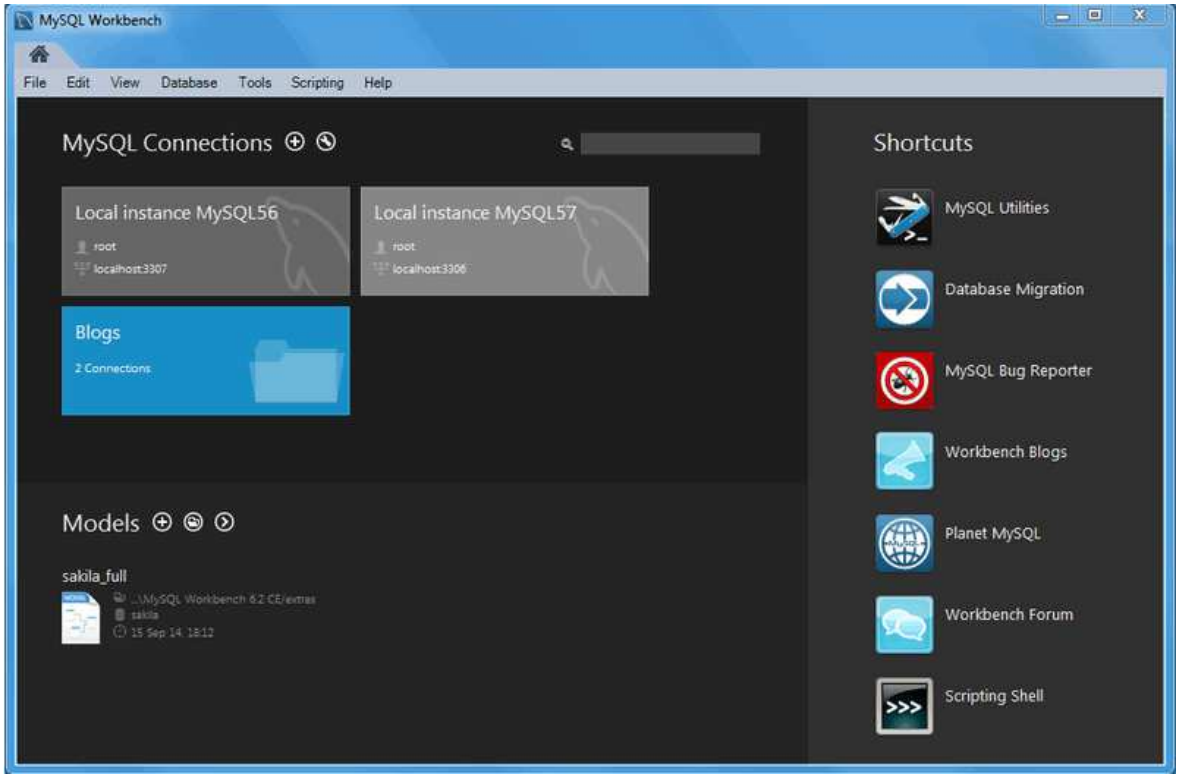
3.4.2 Grafické nástroje

Grafické nástroje MySQL jsou velmi podobné těm, které nabízí i jiné databázové platformy. Mezi nejoblíbenější patří MySQL Administrator. Tento lze ještě využívat, ale vývoj této aplikace byl ukončen a je nahrazen aplikací MySQL Workbench. Tato aplikace poskytuje vývojářům a správcům databází prostředí pro:

- Databázový design a modelování – Umožňuje vytvářet modely schématu databáze graficky.
- Development SQL - Umožňuje vytvářet a spravovat připojení k databázovým serverům. Umožňuje nastavovat parametry připojení, atd.

- Správu databáze – Umožňuje spravovat instance serveru MySQL, provádět zálohy a obnovy dat, kontrolu dat, kontrolovat „zdraví“ databáze a monitorovat výkon serveru MySQL.

- Migraci SQL - Umožňuje migraci z Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL a jiných tabulkách RDBMS, objekty a data do MySQL. Migrace také podporuje migraci z dřívějších verzí MySQL na nejnovější vydání.

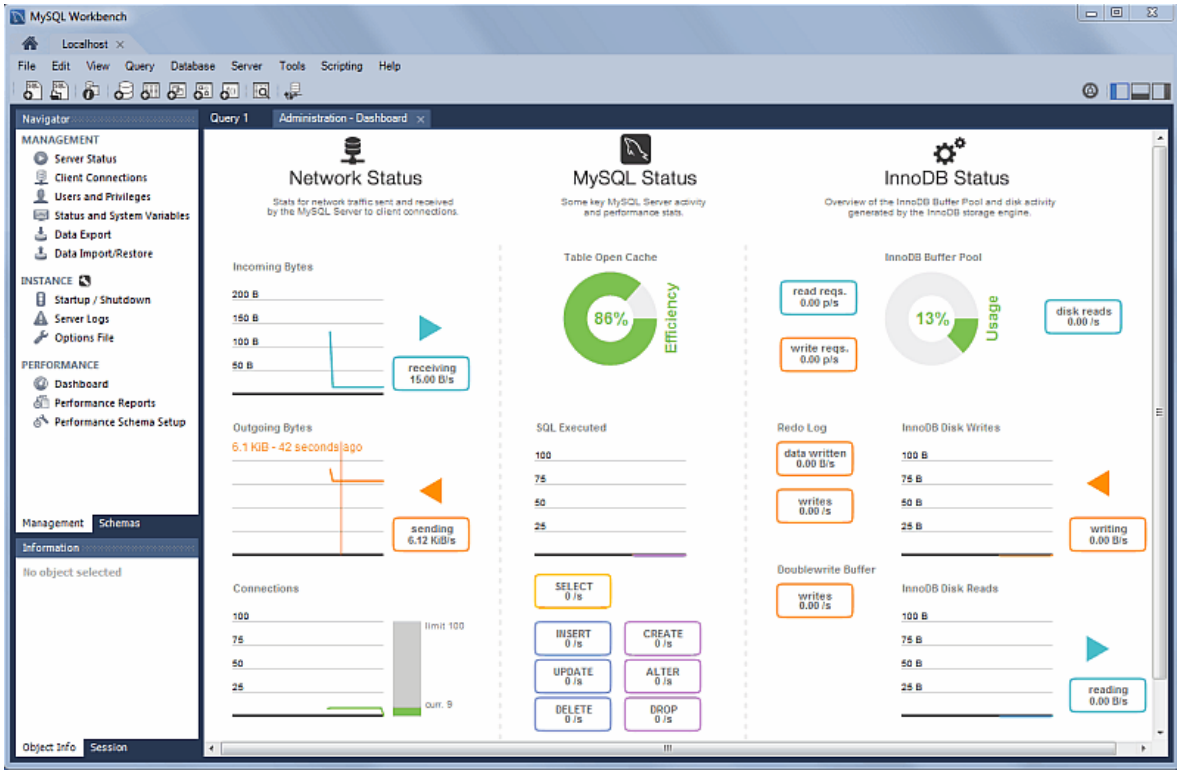


Obrázek 8 - Aplikace MySQL Workbench - Hlavní okno

(Zdroj: http://dev.mysql.com/doc/workbench/en/wb-home.html)

Součástí jsou rovněž nástroje pro optimalizaci výkonu. Jsou to:

- Performance Dashboard



Obrázek 9 - Aplikace MySQL Workbench - Výkon: Dashboard

(Zdroj: http://dev.mysql.com/doc/workbench/en/wb-performance-dashboard.html)

Zde jsou velmi přehledně vidět statistiky Síťového provozu (Network status), který ukazuje velikost odeslaných a přijatých dat MySQL serveru a připojení klientů. Dále pak MySQL Status ukazuje primární činnosti. Počítá (za sekundu) příkazy SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER a příkazy DROP. Poslední InnoDB Status poskytuje přehled o InnoDB bufferu a činnosti disku, který je generován úložištěm InnoDB.

- Performance Schema

Zprávy založené na schématu Performance poskytují reporty operací serveru MySQL prostřednictvím zpráv na vysoké úrovni. MySQL Workbench používá SYS view výkonu generované více než 20-ti reporty, které pomáhají analyzovat výkon databází.. SYS view je dodáván až s verzí MySQL serveru 5.7, ale i starší verze mají vlastní SYS view.

- Query Statistics (statistika dotazů)

Shromažďuje statistické údaje na základě Performance Schema, jako jsou čas vykonávání na straně serveru i klienta, dočasné tabulky, indexy, připojení dalších tabulek (JOIN) a další.

- Visual Explain Plan (vizuální vysvětlení)

Jedná se o grafické znázornění daného dotazu.

3.4.3 Nástroje operačního systému

Výkon lze rovněž sledovat pomocí nástrojů operačního systému. Tyto nástroje však spíše poukazují na příčinu problému.

V případě Windows se jedná o program Správce úloh (Task Manager), který poskytuje informace o využití systému. V systému Windows lze však ještě využít mnohem sofistikovanější konzoly MMC (Microsoft Management Console). Tato má podrobnější informace o aktuální činnosti procesoru, využití paměti i disku. Můžeme zde sledovat velké množství statistik.

Při použití platformy Linux a Unix se nesetkáme s takovým grafickým znázorněním jako ve Windows, ale je třeba upozornit na mnoho nástrojů, komerčních i open-source, které existují. Jejich počet se stále zvyšuje. Red Hat Linux nabízí System Monitor, jako základní program pro sledování výkonu. Společnost IBM vytvořila poměrně sofistikovanou aplikaci rmfpms (Performance Monitor), která je určená pro shromažďování dat a rmf pm pro klientskou aplikaci.

3.5 Návrhy ovlivňující rychlost

Při návrhu databáze nabízí MySQL velké množství funkcí a možností.

3.5.1 Výběr druhu úložiště

Pro výběr druhu úložiště jsou v MySQL minimálně dvě možnosti. První z nich je zadat druh úložiště jako součást jazyka DDL. Příkladem by bylo:

```
CREATE TABLE INNODB_EXAMPLE (FIELD1 INT, FIELD2 INT) ENGINE = INNODB;
```

Druhou možností je zadání v editoru tabulek. Tím může být třeba program MySQL Workbench.

3.5.1.1 Engine MyISAM

Jedná se o výchozí úložný engine MySQL až do verze 5.5. MyISAM ukládá tabulku ve třech souborech. Je to .MYD, což je datový soubor, dále pak .MYI, což je indexový soubor a .frm, kde je struktura tabulky. Formát MyISAM je platformově neutrální, čili datové a indexové soubory lze bezproblémově kopírovat mezi servery založených na různých platformách.

Tabulky mohou na základě definice obsahovat dynamické, či statické řádky. Maximální počet řádků tabulky je limitován velikostí volného místa na disku databázového serveru a také na operačním systému, který omezuje maximální velikost souboru.

Vzhledem k tomu, že MyISAM je nejstarší z úložných enginů (pokračovatel úložiště ISAM) obsahuje i velké množství funkcí, které se v průběhu let postupně vyvíjeli ve prospěch uživatelů. Jedná se především o funkce zajišťující:

- Uzamykání a souběžnost – MyISAM neuzamyká řádky, ale celé tabulky. Užitečnou funkcí je však souběžné vkládání, které umožňuje vkládat nové řádky do tabulky i v době, kdy nad tabulkou běží výběrové dotazy.
- Automatická a ruční oprava – MySQL obsahuje automatickou kontrolu a opravu tabulek MyISAM. Zároveň lze také pomocí příkazu CHECK TABLE nazev_tabulky zkontrolovat zda neobsahuje chyby, případně pomocí příkazu REPAIR TABLE nazev_tabulky opravit chyby.
- Indexové funkce – Tabulky MyISAM obsahují sofistikované indexové funkce, včetně podpory sloupců BLOB a TEXT, které lze vytvářet na 500 znacích těchto sloupců.
- Pozdržené zápisy klíčů – Silně zvýšit výkon lze také u velmi často používaných tabulek pozdrženým zápisem klíčů. Volba DELAY_KEY_WRITE zajistí, že se nebudou na konci dotazu zapisovat data na disk, ale uloží se do bufferu klíčů umístěného v paměti. Na disk jsou pak zapsány až při pročišťování bufferu, nebo při uzavření tabulky.

3.5.1.2 Engine InnoDB

Od verze MySQL 5.5 je Inno DB výchozím úložným enginem. Navržen byl pro zpracování mnoha krátkodobých transakcí, které se obvykle potvrzují a v minimálním počtu případů anulují. Je velmi výkonný s možností automatického zotavení po havárii. Tím je oblíben i pro potřeby netransakčních úložišť. Svá data ukládá do jednoho či několika datových soborů nazývaných tablespace (tabulkový prostor).

InnoDB využívá MVCC (Multiversion concurrency control), kvůli dosažení vysoké souběžnosti. Je využívána strategie uzamykání příštího klíče, aby zabránila fantomovému čtení.

Další populární funkcí, kterou InnoDB nabízí je omezení cizích klíčů. Dále poskytuje velmi rychlé vyhledávání podle hodnot klíče u dotazů, která používají primární klíč. Další funkcí je prediktivní čtení napřed, kde se do zásoby načítají napřed data z disku. Dále pak adaptivní hashovaný index, který vytváří hashované indexy v paměti například pro rychlé vyhledávání s čímž se urychluje vkládání řádků.

3.5.1.3 Engine MyISAM Merge

Jedná se o variantu MyISAM. Tabulky Merge jsou v podstatě zkombinovány z několika tabulek MyISAM do jedné virtuální. Tento engine má několik nevýhod, které jsou navázány na použití úložiště MyISAM a to, že všechny tabulky musí mít stejnou strukturu, provádění značného počtu indexů při zpracování způsobených počtem podtabulek apod. Přesto pro aplikace, které zaznamenávají data do logů, nebo pro aplikace skladového hospodářství se tento engine ukázal jako výhodný, neboť nedochází ke zbytečnému zpracování.

3.5.1.4 Engine Memory

Toto úložiště je známé také z minulosti, kdy bylo nazýváno Heap. Toto úložiště umožňuje vytvářet vysokorychlostní tabulky uložené v paměti, které jsou však k dispozici pouze po dobu běhu serveru MySQL, čili jsou výhodné pro data, která se buď nemění, nebo nevadí jejich ztráta při restartu. Typickým využitím by mohlo být zpracování mezivýsledků při analýze dat, nebo pro vyhledání v tabulce, která mapuje PSČ a názvy měst.

3.5.1.5 Ostatní enginy

V MySQL existuje ještě mnoho úložných enginů, nicméně již nejsou tak využívané jako výše uvedené. Jsou to například ještě:

- Archive – podporuje pouze dotazy INSERT a SELECT, nepodporuje indexy. Dotaz SELECT prochází celou tabulkou, čili využití je pro logování a získávání dat, kdy analýzy prochází celou tabulkou.
- CSV – umí zacházet s csv soubory jako s tabulkami, ale rovněž nepodporuje indexy. Využití by tedy bylo pro výměnu dat mezi programy.
- Federated – neukládá data lokálně, ale odkazuje na nějakou tabulku na vzdáleném MySQL serveru.
- Ostatními jsou pak Blackhole, Falcon, solidDB a další.

3.5.2 Optimalizace struktury tabulky

Při zakládání či změně tabulky je v MySQL možné správným nastavením zvýšit výkon.

3.5.2.1 Zadání formátu řádku

V MySQL lze využít třech typů ukládání řádků a to v pevném, dynamickém a komprimovaném formátu.

Řádky v pevném formátu mají stejnou velikost a tak i mají rychlejší odezvu při přístupu, protože pro databázi to znamená méně práce. Výhodou je menší pravděpodobnost poškození dat. Nevýhodou však je velikost zabraného místa na disku.

Tabulky zadané v dynamickém formátu jsou samozřejmě méně náročné na místo, ale hrozí riziko fragmentace a tím i poškození dat. Pokud však tabulka obsahuje sloupce typu TEXT, BLOB nebo VARCHAR, je automaticky zvolena MySQL dynamická forma.

Komprimované tabulky zabírají nejméně místa z uvedených typů. Rovněž i dotazy jsou rychlejší. Spouští se příkazem myisampack (pro úložiště MyISAM) a po zkomprimování tabulky zůstává pro čtení.

3.5.2.2 Zadání velikosti tabulky

V tomto případě může dojít u velkých dynamických tabulek k nepatrnému zrychlení (v jednotkách procent). Pokud jsou známé hodnoty průměrné délky řádku (AVG_ROW_LENGHT) a očekávané velikosti (MAX_ROWS), pak pomáhají databázi optimálněji vytvářet indexy a dotazy nad takovouto tabulkou jsou rychlejší.

3.5.2.3 Využití indexů

Indexy, neboli klíče jsou datové struktury napomáhající efektivnímu získání dat z tabulky. Ačkoliv jsou pro výkon nezbytné, přesto jsou velmi často opomíjeny. Jejich nezbytnost roste s výší počtu záznamů v tabulce.

V MySQL existuje několik typů indexů. I mezi vývojáři se velmi často mluví o indexech bez přesnější specifikace typu. Jedná se o:

- B-Tree indexy – nejběžnější indexy, používající pro ukládání svých dat datovou strukturu B-Tree
- Hashové indexy – jsou vybudovány na hashové tabulce. Užitečné jsou pouze při přesných vyhledáních, které používají všechny sloupce indexu
- R-Tree indexy – jedná se o prostorové indexy určenými např. pro geografická měření

- Fulltextové indexy – hledá v textu klíčová slova, tzn. neporovnává hodnoty s hodnotami indexu

3.5.2.4 Komprimace klíčů indexu

V případě použití volby, aby databáze komprimovala klíče indexu, dojde k úspoře místa na pevném disku. Zároveň se obvykle sníží rychlost zápisu, ale zvýší rychlost čtení.

3.5.2.5 Typy sloupců

MySQL podporuje mnoho různorodých datových typů, které mohou mít významný podíl na získání, či ztrátě výkonu. Nebudeme se nyní věnovat popisu jednotlivých datových typů, ale je několik bodů, na které by se při volbě datového typu měl brát ohled.

Obecně platí, že z hlediska výkonu je nejvýhodnější použít nejmenší datový typ, který bude reprezentovat uložená data. Čím menší datový typ je použit, tím méně cyklů potřebuje CPU ke zpracování a proto jsou rychlejší. Samozřejmě, že rovněž zabírají méně místa na disku, v paměti i CPU cache.

Rovněž je třeba dbát na jednoduchost datového typu. Kupříkladu porovnání celých čísel je mnohem rychlejší, než porovnání znaků, které komplikují různé znakové sady a kolekce. Velmi častým případem je ukládání IP adresy a datumu (příp. datumu a času) jako řetězce, ačkoliv pro datum má MySQL několik datových typů a IP adresu je výhodnější uložit jako celé číslo.

Posledním nešvarem je časté používání hodnoty NULL. Pokud je to možné je výhodné definovat sloupce s atributem NOT NULL. Spousta tabulek v databázích internetových obchodů je plná hodnot NULL a to jen z důvodu, že se jednalo o výchozí nastavení. Pro databázi nebývá potřeba ukládat informace, že není hodnota k dispozici. Samozřejmě to může být v některých situacích žádoucí, ale přesto by se tímto mělo šetřit. Důvodem je složitější optimalizace dotazů, které odkazují na sloupce s hodnotou NULL. Komplikuje to indexy, nebo porovnání hodnot. Rovněž sloupec, do něhož se může ukládat NULL potřebuje více místa na disku. Ve většině případů lze hodnotu NULL nahradit (prázdným řetězcem, nulou, či speciálním znakem).

Při rozhodování o vhodnosti použití daného datového typu by mělo dojít tedy ve dvou krocích. V prvním vybrat obecnou třídu datového typu (řetězec, číslo, datum a čas, …) a v druhém teprve specifikovat jaké hodnoty mohou nastat, či jaké hodnoty jsou očekávány (VARCHAR, CHAR, TEXT, …).

3.5.2.6 Normalizace

Reprezentace dat v databázi je možná od podoby normalizované po podobu denormalizovanou, včetně jakéhokoliv stavu mezi. Zjednodušeně lze říci, že v normalizované databázi je každá informace pouze jednou a v denormalizované se mohou opakovat. Zdrojů zabývající se normalizací je mnoho. Nyní spíše shrneme výhody a nevýhody normalizované, denormalizované a smíšené varianty výše uvedených.

Normalizované schéma je výhodné hlavně v případě, kde většinu pracovní zátěže tvoří zápis. Důvody jsou následující:

- Normalizované aktualizace jsou většinou rychlejší než denormalizované
- V případě správného vytvoření normalizace nejsou žádná duplicitní data, což je výhodné hlavně v případě změn
- Tabulky jsou menší
- Většinou není třeba v dotazech využívat příkazů DISTINCT a GROUP BY pro odstranění duplicit. Obvykle lze použít triviální dotaz, neboť se tyto hodnoty u normalizované tabulky nachází zvlášť

Nelze však nezmínit i nevýhody, které normalizované schéma s sebou nese. Nevýhody se většinou vztahují k získávání dat z databáze. Pro každý netriviální dotaz je třeba vytvořit alespoň jedno spojení tabulek. Těch však může být mnoho, což způsobuje nepřehlednost dotazu, může to být náročné na výkon a v neposlední řadě to může negativně ovlivnit některé indexové strategie

Denormalizované schéma v mnoha případech funguje velmi dobře. Není potřeba spojovat tabulky, neboť data jsou shromážděna v jedné tabulce. Pro většinu dotazů je nejnáročnější průchod celou tabulkou (i v případě použití indexů). Pokud však se data vejdou do paměti, může to přesto být rychlejší než spojování tabulek. Příkladem, kde je výhodnější nenormalizovat, by mohl být výběr nejnovějších zpráv vložených administrátory. Předpokládáme, že by byly zprávy v jedné tabulce a uživatelé v druhé, v nichž by byla i role (administrátor).

Smíšení normalizovaného a denormalizovaného schématu tak může být tou nejefektivnější cestou. Je prokázáno, že každé schéma má své výhody i nevýhody. Způsobem denormalizace je duplikace vybraných sloupců z jedné tabulky do druhé. Tyto lze od verze 5.0

aktualizovat pomocí triggerů, takže jejich implementace a údržba je podstatně snazší. Lze říci, že duplikované sloupce působí jako speciální cache. U denormalizovaného a smíšeného schématu je však důležité mít na paměti, že je třeba v případě aktualizace udržovat více tabulek.

4 Vlastní řešení

Vlastní práce je založena na praktických zkušenostech autora, v oblasti optimalizace databází a zvýšení rychlosti odezvy internetových obchodů na straně klienta.

Založení vlastního internetového obchodu již dávno není o najmutí programátora, se kterým bude zákazník spolupracovat, aby společně vytvořili fungující e-shop. Dnes si jej může dovolit kdokoliv, neboť internet je plný nejrůznějších řešení a to i bezplatných, kam si i naprostý amatér zadá produkty. V případě malých IT kvalit ručně, v případě zručnějšího zákazníka a uživatele lze naplnit dávkově.

Veškeré optimalizační postupy, které budou níže popsány jsou založeny na autorově praxi ve spojení s teoretickými poznatky v oblasti optimalizace relačních databází. Všechny tyto obchody museli být optimalizačními procesy podrobeny vzhledem k vysoké míře nespokojenosti provozovatelů, neboť procesy byly velmi neefektivní, složité a v konečném důsledku časově náročné.

4.1 Zhodnocení aktuálního stavu internetových obchodů

Úroveň a kvalita internetových obchodů je velmi rozdílná. Velký vliv na to má jednoduchost nasazení. Proto se na internetu objeví i e-shopy, které jsou velmi neprofesionální po stránce nastavení databáze, ale i z hlediska přívětivosti ovládání a grafického zpracování. Vše je do jisté míry závislé také na tzv. poslání daného obchodu.

V dnešní době drtivá většina majitelů produktivních, či funkčních e-shopů vlastní rovněž tzv. kamenné prodejny. Internetový obchod pak slouží primárně jako usnadnění pro své zákazníky, nebo jako podpora prodeje. Zde není tedy kladen takový důraz na jeho kvalitu a obvykle i počet produktů a sortimentních skupin bývá nízký. Pod pojmem nízký můžeme chápat jednotky tisíc produktů a jednotky až desítky sortimentních skupin.

Druhou skupinou vlastníků e-shopů jsou internetové obchody uživatelů, kteří nevlastní žádné zboží a v podstatě jen přeprodávají zboží jiných dodavatelů. Obvykle se jedná o práci při mateřské dovolené, případně vedlejší činnost při hlavním pracovním poměru apod.

Poslední skupinou jsou internetové obchody, jejichž prodej je hlavním a jediným příjmem celé společnosti. Zde je na kvalitu kladen velmi vysoký důraz. Zároveň tyto obchody obsahují velké množství dat a zde již optimalizace všech procesů hraje velmi významnou roli. Není třeba se nyní zaměřovat na největší české e-shopy (Alza.cz, Mall.cz, apod.), které nevyužívají

databáze MySQL a mají vlastní specialisty pro vývoj. Omezíme-li se pouze na menší hráče na trhu, které spojuje hned několik požadavků. Jsou to:

- Nízká výrobní cena
- Brzké nasazení
- Jednoduché prostředí z pohledu zákazníka (přátelský a intuitivní vzhled)
- Přívětivý backend (prostředí pro správu) pro majitele, či správce dat
- Vysoká rychlost načítaných dat

Tyto obchody by dle výše uvedené kategorizace patřili do první a poslední skupiny. Zhruba 90% těchto obchodů produkuje hrubé roční tržby do 10 miliónů korun českých. Optimalizace takových obchodů má velký smysl. Na základě kontaktů s provozovateli bylo vytyčeno několik hlavních důvodů k optimalizaci. Jsou to:

- Velká konkurence – Obchodů v téměř všech sortimentních oblastech jsou na českém internetu desítky až stovky a cenové rozdíly jsou minimální. Je tedy třeba vyjít zákazníkovi vstříc, aby se mu v aplikaci dobře pracovalo a nechtěl ji měnit.
- Závislost na tržbách – Provozovatelé jsou závislý na příjmech z internetových prodejů. Je tedy opět nutné, aby zákazník byl spokojený a na internetový obchod se vracel.
- Rozšíření produktové nabídky – Tento jev nemá vliv na optimalizaci. Je třeba pouze sledovat množství dat, aby nezpůsobila aplikaci problémy, ale to se obvykle nestává.

4.2 Optimalizace databáze ve spojení s frameworkem Zend

První optimalizační procesy databáze ve spojení s frameworkem Zend jsou více než nutné. Zend Framework patří co se rychlosti a doby odezvy týká mezi ty nejpomalejší. Ve spojení se špatně vytvořenou databází se internetový obchod stává téměř nepoužitelným. Práce se bude velmi často odkazovat na web *jeany.cz*. Tento web byl autorem kompletně optimalizován, neboť po cca. ročním používání byl z důvodu latence odezvy minimálně využíván zákazníky a majitelé byli odkázáni pouze na tržby z kamenného obchodu, ačkoliv zaměření a název domény by mohl předurčit k většímu využití ze strany zákazníků a tím pádem i k vyšším tržbám. Vzhledem k relativně vysokému vstupnímu nákladu za internetový obchod

a téměř nulové návratnosti bylo požadováno pouze „zrychlení“ bez změny databáze a frameworku, neboť další změny by vedli k navýšení ceny.

Zvoený postup zahrnoval standardní optimalizační metody:

- Využít výhod frameworku Zend ve prospěch celé aplikce
- Seznámit se a případně optimalizovat databázový server (případně servery) a využít plně dostupný hardware
- Začít s laděním databáze

4.2.1 Použití frameworku Zend

Ačkoliv Zend, jak je uvedeno výše, patří mezi nejpomalejší frameworky, tak je potřeba také zmínit, že existuje mnoho dobrých důvodů, proč jej použít. Chybou však může být využití hostingu, který nevlastní Zend Optimizer nebo Accelerator. Tento nástroj může zrychlit aplikaci v řádu desítek procent.

Obvykle lze optimalizovat již na úrovni Zend Frameworku, například odpojením od externích nástrojů. Toto se však netýká optimalizace databáze, takže není důvod zde toto téma dále rozvádět.

4.3 Optimalizace databázového serveru

Nastavením databázového serveru lze dosáhnout jistých výkonnostních zisků. MySQL server ve výchozí konfiguraci nepoužívá mnoho zdrojů. Záměrem je totiž, aby byl všestranný a proto se nepředpokládá, že by na serveru, kde je nainstalovaný byl provozován pouze tento databázový server. Ve výchozí konfiguraci se tak počítá jen se zdroji potřebnými pro nastartování MySQL a se spouštěním jednoduchých dotazů nad malými objemy dat (v řádu MB dat).

Obvykle však aplikace, na které je tato práce zaměřena (a bylo tomu např. i u modelového e-shopu jeany.cz) běží na jednom stroji databáze, Apache i PHP. Využívají tedy všechny společné prostředky (disky, RAM,…). Pravděpodobně nejdůležitějším prostředkem je RAM. Vzhledem k tomu, že o RAM paměť se dělí všechny programy tzv. triády, je důležité, aby RAM paměti bylo dostatek. Na druhou stranu, pokud RAM paměť máme, tak by měla být využitá.

V této oblasti je velmi výhodné použít skript MySQLTuner. Jedná se o skript napsaný v Perlu, který do verze MySQL 5.7 má plnou podporu. Tento skript dokáže pomoci s konfigurací MySQL a lze z něj získat doporučení pro zvýšení výkonu a stability.

Obrázek 10 - MySQLTuner - ukázkový výpis

(Zdroj: http://mysqltuner.com/)

4.4 Spuštění Slow Query Log

Logování pomalých dotazů je jednou z nedílných součástí kroků před optimalizací databáze. Je doporučováno toto logování provádět v několika krocích kvůli přehlednosti. Poprvé je vhodné použít výchozí nastavení, které činí 10 sekund. Po vyhodnocení log reportu se obvykle snižuje čas na menší hodnoty. Příklad nastavení logování je:

```
long_query_time = 5
log_slow_queries = /var/log/mysql_slow_queries.log
```

Výše uvedený příklad zapsaný v konfiguračním souboru MySQL (/etc/my.cnf) nám bude logovat všechny dotazy trvající déle jak 5 sekund. Samozřejmě logování má smysl tehdy, je-li databáze zatěžovaná.

Z logu pak lze vyčíst dotazy, které jsou pravděpodobně bez indexů, nebo například dotazy, které obsahují vnořené výběrové dotazy (dotazy `SELECT`).

Velmi častým případem, který se vyskytuje u mnoha e-shopů, je postupné doplnění dotazů vlastními funkcemi programátorů. V mnoha případech se právě lze setkat se zbytečně dlouhými dotazy s několika vnořenými dotazy SELECT, či velmi zbytečně napojenými tabulkami pomocí příkazu JOIN (tento případ je častější).

4.5 Výběr správného úložiště

Jak bylo popsáno v teoretické části, základním stavebním kamenem je výběr správného úložiště. Primárně jde o rozhodnutí mezi MyISAM a InnoDB. Pro malé e-shopy je rozdíl téměř nepostřehnutelný, ale v případě zvýšeného počtu zápisů už rozdíl lze vidět a změřit. Pokud však budeme trvat na teoretických principech, pak bychom se měli zamyslet nad počtem transakcí. InnoDB je pomalejší než MyISAM, ale v případě většího počtu transakcí již rozdíl můžeme naměřit. Přesto lze říct, že problém se špatně zvoleným úložištěm MyISAM vs. InnoDB by ve velikosti internetového obchodu, na který se zde zaměřujeme byl velmi výjimečný. A to ani v případě velkých importů, neboť ty jsou prováděny v době provozního klidu (noční hodiny), takže chyba by neměla na funkčnost sebemenší vliv.

Přestože nemusí nutně přinést volba úložiště výkonnostní zisk, je nutné vzít v potaz vlastnosti těchto úložišť. Výkonnostní benefit může MyISAM přinést tam, kde výrazně převyšuje operace čtení nad operací zápisu. Lze říci, že měřitelný zisk lze zaznamenat, pokud bude operace čtení tvořit cca. 85% nad 15% operací zápisu.

Ačkoliv literatuře nenajdeme přesná doporučení, aktuálně preferovaným je využívání enginu InnoDB. Jednak již podporuje fulltextové vyhledávání (od verze 5.6), časové ztráty i v případě vysokého procenta čtení jsou minimální, ale hlavně neohrožuje integritu dat. Pokud jde o fulltextové vyhledávání, je doporučováno toto řešit mimo databázi a využít nástrojů, které jsou pro toto vyvinuty.

Pro potvrzení této teorie bylo spuštěno 100 dotazů pro vyhledání zboží dle indexovaného pole v tabulce s 10.000 záznamy. Rozdíl v součtu dotazů je v řádu desetin vteřin, což je opravdu zanedbatelné.

4.6 Použití příkazu EXPLAIN

Tento příkaz patří mezi speciální příkazy, které pomáhají vývojáři optimalizovat SQL příkazy, aby byly rychlejší. Použití dotazu je velmi jednoduché. Před klíčové slovo SELECT stačí vložit EXPLAIN. Pro kvalifikované posouzení je třeba výsledky správně interpretovat. Velmi jednoduchým, názorným příkladem může být následující:

```
EXPLAIN SELECT * FROM zakaznici;
```

Výsledkem by bylo:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	zakaznici	ALL	*NULL*	*NULL*	*NULL*	*NULL*	4710	

Obrázek 11 - Příklad příkazu EXPLAIN (1)

(Zdroj: Autor)

Výsledek lze interpretovat následovně. Dotaz neobsahuje žádné další poddotazy ani uniony (select_type = SIMPLE), dotaz přistupuje k tabulce zakaznici (table = zakaznici), projde celou tabulkou, než najde záznam (type = ALL), pro dotaz v podstatě nejdou použít indexy (possible_keys = NULL), dále MySQL se nerozhodl použít žádný klíč pro optimalizaci (key = NULL) a logicky pak nelze ani ukázat kolik bajtů se MySQL rozhodl použít pro index (key_len = NULL), nemáme ani žádné reference na předchozí tabulky, neboť dotaz obsahuje pouze jednu (ref = NULL), musí přečíst 4710 řádků (rows = 4710) a nemáme žádnou informaci navíc od MySQL (Extra = *prázdné*).

Trochu zajímavější a již více říkající by mohl být rozbor příkazu, který je použit níže:

```
EXPLAIN SELECT p.nazev FROM produkty AS p LEFT JOIN znacky AS z
ON z.znacka_id = p.znacka_id WHERE z.nazev ="Diesel";
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	z	ALL	PRIMARY	*NULL*	*NULL*	*NULL*	10	Using where
1	SIMPLE	p	ref	fk_zbozi_znacky1	fk_zbozi_znacky1	4	jeanycz.z.znacka_id	431	

Obrázek 12 - Příklad příkazu EXPLAIN (2)

(Zdroj: Autor)

Interpretace tohoto příkazu by byla, že dotaz neobsahuje poddotazy, skládá se ze dvou tabulek (zde jsou názvy programátora, nikoliv názvy z databáze), přičemž tabulku „z“ projde celou, ale v tabulce „p“ již hledá podle hodnoty klíče indexu. Vidíme, jaké indexy se mohou v tabulkách použít a kolik paměti zabírají. Je vidět jaká hodnota indexu bude použita k vyhledání hodnot a kolik řádků bude muset projít.

Podrobný popis možných typů lze dohledat v oficiální dokumentaci MySQL.

4.7 Využití indexů

V oblasti středně velkých internetových obchodů lze většinu problémů s pomalými dotazy vyřešit správným indexováním. Účelně je zde použito slovo „správné“ indexování, neboť mnohdy se můžeme setkat s databázemi, které mají indexováno příliš mnoho atributů.

Indexovány by měly být všechny sloupce, podle kterých se vyhledává, třídí, nebo pomocí kterých se spojují tabulky v dotazech. Ačkoliv u tabulek v řádu desítek řádků (i jednotek stovek) nemá indexování větší smysl, ale u větších tabulek se můžeme již setkat s výkonovými problémy, pokud nastavení indexů bude zanedbáno.

Na druhou stranu je třeba brát na zřetel, jak indexy fungují. Vytvořením indexu dojde k rezervaci části paměťového prostoru, kam je následně uložena informace o rozmístění hodnot indexovaných sloupců v tabulce. Tvorbou indexů se tak navyšují nároky databázového serveru na operační paměť.

Na již několikrát uvedeném e-shopu jeany.cz jsou pouze 4 vyhledávací parametry pro zboží a 2 parametry třídění, čili indexace hodnot nebude působit větší problémy. Podstatně větší problémy nám může způsobit internetový obchod zabývající se výpočetní technikou, kde při výběru např. notebooku může být vyhledávacích parametrů podstatně víc (více jak 10), nebo internetový obchod prodávající např. ojetá vozidla. Zde již musí programátor velmi poctivě řešit využití paměti a za zvážení by stálo využití dalšího nástroje, příp. použití aplikace, která s indexy umí pracovat lépe než MySQL.

4.8 Rozdíly ve výkonu mezi JOIN a vnořenými dotazy SELECT

Velmi často mohou způsobit výkonnostní problémy i vnořené dotazy. Vždy je nutné zvážit, zda takový příkaz je nutný. Velmi výhodné totiž bývá tyto dotazy nahradit spojením tabulek pomocí JOIN. Důvod je jednoduchý. V případě vnořených SELECTů se neuplatňují indexovaná pole ve vnořeném dotazu a to i v případě, že jsou nastaveny.

Konkrétní příklad může být, výběr produktů značky Diesel. V tabulce produktů je k dispozici pouze ID značky a je tedy nutné spojit s tabulkou značek. V tabulce značek je 10 záznamů a v tabulce produktů je 8.800 záznamů. Výsledkem bude výpis 2.300 záznamů. Nejprve spustíme původní dotaz, který byl generován programovým kódem a obsahoval vnořený dotaz SELECT:

```
SELECT p.nazev FROM produkty AS p WHERE p.znacka_id = (SELECT z.znacka_id FROM znacky AS z WHERE z.nazev ="Diesel");
```

Tento příkaz trval 0,0015 s, zatímco níže uvedený příkaz trval 0,0007 s, čili byl proveden o polovinu rychleji.

```
SELECT p.nazev FROM produkty AS p LEFT JOIN znacky AS z ON z.znacka_id = p.znacka_id WHERE z.nazev ="Diesel";
```

Výše uvedený případ byl zobrazen na velmi triviálním příkladu, ale i z toho je patrné, že v případě použití nad složitějším výběrem může způsobit výkonnostní problém. Obvykle tyto dotazy jsou programově generovány použitím cyklů (např. cyklus FOR), takže by mohlo dojít k vícenásobnému vnoření.

4.9 Optimalizační techniky na příkazem INSERT

Příkaz INSERT je jedním ze základních příkazů. Nicméně i tento lze významně optimalizovat, byť záleží na typu obchodu. V jistých případech lze velmi často používané příkazy INSERT nahradit jedním, hromadným příkazem INSERT. Samozřejmě, že je časový rozdíl, pokud vkládáme jeden záznam po druhém, nebo několik hromadně.

```
INSERT INTO kosik (ID_art, name_art) VALUES (123675, M955 257 630 009 Billstrong);

INSERT INTO kosik (ID_art, name_art) VALUES (123677, Hudson 403);
```

nebo

```
INSERT INTO kosik (ID_art, name_art) VALUES (123675, M955 257 630 009 Billstrong), (123677, Hudson 403);
```

Jak je uvedeno výše do jisté míry záleží na typu e-shopu. U obchodu zabývajícího se primárně prodejem kalhot nemá tato optimalizace velký smysl, neboť se nedá předpokládat, že by zákazník objednával velké množství produktů. Nicméně např. obchody zabývající se stavebnictvím a stavebními pracemi obsahují velké množství komodit na jedné objednávce.

Příkladem by mohl být také obchod s instalatérským materiálem, kde většina objednávek obsahuje větší množství položek. Stejně tak stavebniny, kdy odběratelem jsou drobní živnostníci, či soukromé osoby. V takovém odvětví zákazníci nakupují zboží, které budou potřebovat později, neboť ve většině případů nepodléhá zboží zkáze a kvůli ceně potřebují doložit vozidlo zajišťující dopravu.

4.10 Optimalizace pro srovnávací servery

Při optimalizaci je třeba brát také na zřetel napojení databáze a aplikace na servery třetích stran. Typickým případem může být napojení na srovnávací server (např. Heureka.cz). Tyto servery fungují obvykle na základě specifických XML souborů. Většina e-shopů tak na dotazy předává XML soubor v reálném čase, což může za jistých okolností být nevýhodné. Můžeme se tedy setkat se zpomalením právě z důvodu, že server se dotázal v nevhodnou dobu.

Řešením, které je velmi často aplikováno je ve formě předgenerovaných XML souborů. V době dotazu, tak nedochází k žádnému zatížení databáze, neboť server obdrží již dříve předgenerovaný XML soubor. Ten se generuje asynchronně, takže využívá prostředků v době, kdy jsou k dispozici.

5 Zhodnocení výsledků

Jedním z nejdůležitějších úkonů při optimalizaci databáze a to bez ohledu zda se jedná o MySQL, či jinou je již zpočátku znát veškeré požadavky zadavatele. V podstatě totéž platí i pro vytváření nové databáze, kde by vývojář navíc měl pracovat i s určitou predikcí v oblasti počtu komodit, zákazníků i tržeb. Současné internetové obchodování v České republice je na vzestupu, stejně jako se zvyšuje gramotnost uživatelů přistupujících k internetu za účelem nákupu. Podle této tendence se musí také logicky zvyšovat nároky kladené na tyto aplikace.

Dle výsledků patří mezi nejčastější chyby v databázích internetových obchodů špatné indexování. V podstatě všechny internetové obchody jsou více či méně založeny na vyhledávacích polích a filtrech. Správné indexy umožní tak získat data velmi rychle. Na druhou stranu je třeba brát zřetel, aby databáze nebyla přeindexovaná a indexy nezabírali příliš mnoho místa. V tomto směru stojí za zvážení využití dalšího nástroje, který tyto problémy odstraní. Jedním z takových nástrojů může být například Elasticsearch, což je fulltextový vyhledávač vyvinutý v Javě a založený na Apache Lucene.

Hlavní výhodou bezplatného produktu Elasticsearch je jednak vysoká rychlost. Ta se dá využít právě třeba při nejrůznějších vyhledáních. Není nutno nijak vstupovat do databáze, ale výsledky vyhledání produktu můžeme rovnou uživateli zobrazit (pokud tam umístíme například i popis). Ten pak již výběrem konkrétního produktu vybere detailní informace o produktu, ale zde již aplikace zasílá do databáze velmi jednoduchá příkaz (SELECT) se známým ID. Ačkoliv zde nejde o přímou optimalizaci databáze, tak lze zcela jistě tvrdit, že jde o optimalizaci celé aplikace a ulehčit databázi, která může vykonávat jiné činnosti.

Obecně lze říci, že alfou a omegou internetových obchodů menších velikostí je správné indexování a správné sestavení dotazů. MySQL v tomto směru nabízí dostatek sofistikovaných nástrojů, které mohou vývojářům pomoci chyby odstranit.

Velmi důležitým ukazatelem je také log dlouhých dotazů, neboť ve velmi hojné míře se lze setkat s neoptimalizovanými dotazy, které vyhledávají dle neindexovaných polí, či dotazy jsou velmi špatně napsané, či programově doplňované. Tyto problémy vznikají jednak doplněním vlastních funkcionalit a tím i rozšířením dotazů do volně stažitelného e-shopu, nebo změnou programátorů na dané aplikaci. Nejdůležitějším měřítkem při budování a úpravě e-shopu totiž ze strany provozovatele bývá cena. Programátor tak neřeší úpravy v kontextu

s vytvořeným obchodem, ale pouze doplněním funkcionality za účelem minimálně stráveného času programováním a tím pádem i snížením svých nákladů.

6 Závěr

Práce byla zaměřena na jednu z nejvíce používaných aplikací na internetu a to internetový obchod. Tato se těší stále větší oblibě uživatelů, a proto dochází k nárůstu těchto obchodů i v současné době. Vzhledem k velké konkurenci je tedy potřeba hledat nejrůznější výhody, které by upřednostnili konkrétní obchod před jiným. Jedním z mnoha řešení je optimalizace databáze. Tato práce popisuje postupy vedoucí k odstranění nejčastějších problémů mající vliv na rychlost internetových obchodů.

Na základě prostudovaných informačních zdrojů z oblasti relační databáze, MySQL, ale také ústní komunikace s provozovateli a zákazníky došlo k vymezení stěžejních cílů potřebných pro efektivní optimalizaci. Práce byla zaměřena na metody relačně databázové technologie.

Pro vyhledání správného řešení bylo třeba zjistit tzv. úzká hrdla aplikace. Na základě teoretických znalostí je nutné získat výsledky nejrůznější výkonnostních testů a ty správně interpretovat. Ze zjištěných poznatků a praktických zkušeností je třeba podotknout, že v oblasti těchto aplikací se jedná primárně o efektivní nastavení na straně databázového serveru, aby celá aplikace měla pro správnou a korektní funkčnost dostatek prostředků.

Práce byla také zaměřena na činnosti, které vývojářům a programátorům pomáhají najít chyby a omezení, které jsou v aplikaci zaneseny, aby mohli být odstraněny. Vzhledem k tomu, že nalezení problému je stěžejním, jsou v práci rozebrány možnosti využití aplikací, pro vyhledání problému na celém systému, či jednotlivých dotazech.

V případě využití v práci popsaných nástrojů lze zjistit chyby a problémy, které aplikace vykazuje a na základě jejich správné interpretace tyto chyby opravit. Tento proces může přinést benefity pro provozovatele v lépe fungující aplikaci a tím i rychlejší odbavení zákazníka, neboť mimo ceny je toto právě stěžejní pro udržení konkurenceschopnosti na přeplněném trhu.

Mezi stěžejní činnosti optimalizace nepochybně patři indexace. Vytvoření a správné nastavení klíčů je dle zjištění velmi opomíjeno a právě toto patří mezi významné činitele dané problematiky.

Je nutné samozřejmě zdůraznit, že pojem optimalizace je velmi úzce spjat s kvalitou. Proto je třeba brát ohled i na to, že pokud by navrhované postupy neměly významný vliv na

rychlost aplikace, neboť mohou obsahovat malé množství dat, tak zcela jistě budou mít vliv právě na kvalitu software.

Proces optimalizace stejně jako databázový systém MySQL patří mezi velmi zajímavá témata a podrobnější studium této oblasti může velmi obohatit znalosti čtenáře. Zároveň je třeba podotknout, že MySQL patří mezi nejoblíbenější a nejrozšířenější open-source aplikace na světě a ačkoliv se jedná o bezplatný systém, lze na něm provozovat i profesionální aplikace velikosti ERP programu.

7 Seznam použitých zdrojů

1. NOSKA, Martin. *Výzkum Googlu: Co a jak Češi nakupují na internetu?*. [online] 8.11.2008. [citace 17.12.2014]. Dostupné z: http://computerworld.cz/internet-a-komunikace/vyzkum-googlu-co-a-jak-cesi-nakupuji-na-internetu-276

2. Marketingové noviny. *Historie elektronických obchodů*. [online] 20.7.2006. [citace 17.12.2014]. Dostupné z: http://www.marketingovenoviny.cz/marketing_4391/

3. BusinessWorld. *Internetový obchod v ČR zatím stále roste, na prvním místě je elektronika*. [online] 17.11.2013. [citace 17.12.2014]. Dostupné z: http://businessworld.cz/analyzy/internetovy-obchod-v-cr-zatim-stale-roste-na-prvnim-miste-je-elektronika-11247

4. PEACOCK, Michael. *Programujeme vlastní e-shop v PHP5*. 1. vydání. Brno: Computer Press, a.s., 2011. ISBN 978-80-251-3181-7

5. WILLIAMS, E. Hugh, LANE, David. *Programujeme webové aplikace pomocí PHP a MySQL*. Brno, Computer Press, 2003. ISBN 80-7226-760-4

6. CONVERSE, Tim, PARK, Joyce, MORGAN, Clark, *PHP & MySQL Bible*. Indianapolis: Wiley Publishing, Inc., 2004. ISBN 0-7645-5746-7

7. SCHWARTZ, Baron, ZAITSEV, Peter, TKACHENKO, Vadim, ZAWODNY, Jeremy D., LENTZ, Arjen, BALLING, Derek J. *MySQL profesionálně – Optimalizace pro vysoký výkon*. 1. vydání. Brno: Zoner Press, 2009. ISBN 978-80-7413-035-9

8. SCHNEIDER, Robert D. *MySQL: Oficiální průvodce tvorbou, správou a laděním databází*. 1. vydání. Praha: Grada Publishing a.s., 2006. ISBN 978-80-247-1516-2

9. HÜBNER, David. *Upgradujeme MySQL server* [online] 28.6.2010. [citace 12.2.2015]. Dostupné z: http://www.zdrojak.cz/clanky/upgradujeme-mysql-server/

10. Oracle Corporation. *MySQL 5.7 Reference Manual* [online] [citace 15.11.2016]. Dostupné z: http://dev.mysql.com/doc/refman/5.7/en/mysqladmin.html