Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Zabezpečení a správa přístupů MySQL

Václav PEKÁREK

! ! !

Místo této strany vložíte zadání bakalářské práce.
(Do jedné vazby originál a do druhé kopii)

! ! !

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Zabezpečení a správa přístupů MySQL" 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 30.3. 2012 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce panu Ing. Martinu Havránkovi za rady, připomínky a odborné vedení při zpracování této práce.

Zabezpečení a správa přístupů MySQL

Securing and managing access to MySQL

Souhrn

Tato bakalářská práce se zabývá zabezpečením a správou přístupů k systému řízení databáze - MySQL, popisuje historický vývoj databází, vznik relačních databází a architekturu serveru MySQL. Tato práce také zahrnuje obecné aspekty zabezpečení a zálohování databáze, včetně konkrétních ukázek a doporučení pro MySQL.

Summary

This bachelor's thesis concerns securing and managing access to Database management system – MySQL, describes historical evolution of databases, creation of relational databases and architecture of MySQL server. This thesis also includes general aspects of security and backup of database, including specific examples and recommendations for MySQL.

Klíčová slova: MySQL, SQL, databáze, zabezpečení, správa přístupu, zálohování.

Keywords: MySQL, SQL, databases, securing, managing access, backup.

Obsah

1. Úvod

Databáze jsou v oblasti informačních technologií velice důležité. Databáze obsahují data o vypůjčených knihách z knihovny, čísla sociálních, zdravotních a bankovních účtů až po encyklopedie a novinové články posledních desetiletí. Databáze utváří úložiště dat, ke kterému přistupuje současně mnoho uživatelů. Kromě samotných dat také obsahuje popis těchto dat, metadata. Obsažená data jsou v databázi reprezentována v logických souvislostech a tvoří mezi sebou vztahy. S databází se pracuje prostřednictvím systému řízení databáze (DBMS[1]), který komunikuje s uživateli, různými aplikacemi a i se samotnou databází. DBMS tvoří významnou část serverů a často i samostatný server. Při použití v rozsáhlých, robustních a kritických systémech je DBMS tvořen celou sítí serverů vzájemně propojených skrze síť Internet.

Systém řízení databáze MySQL, s kterým je pracováno v této práci, je používán celou řadou významných společností (Wikipedia, Facebook, Routers aj.) a je v nabídkách většiny hostingových provozovatelů. Předností MySQL je rychlost, jednoduchost (nastavení a správa databáze je mnohem snazší než u rozsáhlých systémů), standard SQL, kompatibilita s mnoha interfejsy pro práci s dotazy (příkazový řádek, webový prohlížeč, nebo klient s grafickým rozhraním), přenosnost mezi operačními systémy (Unix, Windows, NetWare) a hardwarovými zařízeními (servery, osobní počítače a i přenosná zařízení) a v neposlední řadě také cena (MySQL je šířeno jako Open Source pod licencí GNU General Public License) (DuBois, 2009). Podpora MySQL společně s databázemi PostgreSQL a dalšími je v celé řadě redakčních systémů (Drupal, WordPress, Joomla! aj.), umožňujících uživatelsky jednoduše a kvalitně vytvářet obsah webových stránek.

Pro kombinaci databáze MySQL společně se skriptovacím jazykem PHP a serverem Apache na platformě operačního systému LINUX se vžilo označení LAMP. Takovéto open-sourcové řešení je alternativou ke komerčním produktům a přitom je kvalitní.

Tato práce je rozdělena do dvou částí. První část práce obsahuje teoretická východiska pro porozumění databázím obecně. Obecné poznatky o architektuře databází jsou popsány

[1] DBMS - Database management system, systém řízení databáze.

v časovém kontextu spolu s postupným vývojem jazyka SQL. Tyto poznatky o databázích jsou v práci dále doplněny a konkretizovány pro databázi MySQL. Práce obsahuje teoretická východiska nutná pro porozumění funkce zabezpečení v prostředí databázového serveru na platformě LINUX.

Vlastní práce uvádí na příkladech možnou a bezpečnou konfiguraci databázového serveru pro různé situace nasazení MySQL. Dále se práce zabývá příklady bezpečnostních rizik spojených s využíváním databáze MySQL s aplikacemi v prostředí webových služeb. A také dopodrobna rozebírá problematiku zálohování, respektive zotavení databáze a konzistence dat.

2. Cíl práce a metodika

2.1. Cíl práce

Cílem této práce je popsat databázi MySQL s užším zaměřením na problematiku bezpečnosti a zhodnocení správy přístupů k této databázi.

V praktické části je cílem práce zhodnocení bezpečnostních rizik a nedostatků databáze MySQL a návrh jejich možných řešení.

2.2. Metodika

Práce čerpá ze zdrojů odborné literatury a zdrojů zveřejněných na internetu, uvedených v seznamu použitých zdrojů. V praktické části je čerpáno i z přímých konzultací s techniky provozovatelů webhostingových služeb a z vlastních poznatků s konfigurací MySQL databáze.

V práci se vychází z instalace MySQL serveru verze 5.5 na platformě LINUX / Debian, není-li uvedeno jinak. V práci jsou uváděna nastavení konfiguračního souboru */etc/mysql/my.cnf*.

3. Přehled řešené problematiky

První komerční programy pro práci s rozsáhlými daty (Integrated Data Store), později síťové a hierarchické systémy byly vyvinuty v 60. letech minulého století. V 70. letech došlo s využitím Relačního modelu dat k formování a vzniku databázových tech-nologií, které se rozvíjejí dodnes (Pokorný, 1994, s. 11) Databázové technologie jsou tvořeny systémem řízení báze dat DBMS (Database Management System). DBMS vytváří rozhraní mezi aplikacemi a uloženými daty.

3.1. Architektura databází

Architekturu databází můžeme rozdělit podle jednotlivých vrstev. Každá vrstva obsahuje určitou komponentu, tzn. služby uživatelské, aplikační, datové a samotnou databázi. Jednovrstvá architektura obsahuje pouze centrální systém a k němu připojené zobrazovací terminály. Dvouvrstvá architektura je tvořená klientem a serverem. Rozlišuje se podle umístění aplikačních služeb: u klienta (tzv. „tlustý“ klient), na serveru (tzv. „tenký“ klient). Jsou-li aplikační služby soustředěné u klienta, je kladen velký nárok na přenosovou kapacitu spojení mezi klientem a serverem. Oproti tomu architektura s tzv. tenkým klientem a s aplikačními službami umístěnými na serveru přenáší ke klientovi pouze uživatelské služby a jen vyžádané informace. Třívrstvá architektura postupuje v oddělení služeb ještě dále a rozděluje aplikační služby a datové služby. Tyto samostatné logické celky mohou být na rozdílných serverech, nebo i na stejném serveru. Toto rozdělení umožňuje zvýšení bezpečnosti (podrobněji se tímto zabývá kapitola 3.4 Zabezpečení) a efektivní rozdělení zátěže. Aplikační služby, nesoucí významnou část provozní zátěže, mohou být rozděleny na více serverů (Vysoká škola báňská, 2012; Zendulka, 2012).

3.2. Jazyk SQL

3.2.1. Historický vývoj jazyka SQL

V roce 1974, v laboratořích IBM Santa Teresa v San Jose v Kalifornii, firma IBM započala vývoj projektu System/R. Úspěšné splnění cíle v letech 1974 - 1975 potvrdilo

životaschopnost relačního modelu. Následovalo získávání zkušeností s návrhem a implementací relačních databází. V roce 1974 Donald D. Chamberlin a Raymond F. Boyce, kteří pracovali na projektu System/R, publikovali přelomový článek SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE (Chamberlin, 1974).

Vznikl jazyk SEQUEL, který umožňoval uživateli dotazovat se relační databáze pomocí srozumitelných vět anglického jazyka. Tento jazyk byl poprvé implementován v prototypové databázi SEQUEL-XRM.

V letech 1976 a 1977 D. D. Chamberlin s kolegy pokračoval ve výzkumu jazyka SEQUEL. Původní verzi přepracovali na SEQUEL/2. Z právních důvodů byl jazyk přejmenován na SQL (Structured Query Language - strukturovaný dotazový jazyk). Jazyk SQL umožnil podporu vícetabulkových dotazů, či sdílený přístup k databázi mezi více uživateli. V roce 1979 byl projekt System/R ukončen se závěry o uskutečnitelnosti relačních databází a užitečnosti pro využití jako komerční produkt.

Průběhem 70. let byl System/R společnosti IBM velmi diskutován na odborných seminářích a sledován odbornými časopisy. Jakmile bylo zřejmé, že společnost IBM vážně uvažuje o komerčním potenciálu a vyvíjí produkty založené na relační databázové technologii a SQL, byla odborníky založena společnost Relational Software Inc. v Menlo Park, Kalifornie v roce 1977. Ta měla za cíl sestavit nový relační databázový produkt založený na jazyku SQL. Komerční produkt Relational Database Management System - RDBMS (systém správy relačních databází) byl nazván Oracle a uveden na trh v roce 1979. O dva roky dříve než uvedla na trh komerční produkt společnost IBM. Předností produktu Oracle byla práce na malých počítačích Digital VAX, nikoli na mainframech společnosti IBM. Relational Software Inc. byla přejmenována na Oracle Corporation.

Technologii relačních databází zkoumali Michael Stonebraker a Eugene Wong v počítačových laboratořích univerzity Berkeley. Vyvinuli prototypovou relační databázi INGRES, která obsahovala databázový jazyk QUEL - Query Language (dotazový jazyk). Jazyk QUEL byl oproti SQL více strukturovaný a obsahoval méně příkazů ve stylu anglického jazyka. Když bylo potvrzeno směřování jazyka SQL v tomto oboru k standardnímu databázovému jazyku, byl systém INGRES transformován na systém správy relačních databází s využitím SQL.

V roce 1980 část výzkumného týmu opustila univerzitu v Berkeley a založila firmu Relational Technology. O rok později byla uvedena první komerční verze produktu INGRES (Kofler, 2007; Gilmore, 2007; Valenta, 2012).

3.2.2. Dotazovací jazyk SQL

Strukturovaný dotazovací jazyk vznikl v duchu dodržení konstrukce přirozených otázek anglického jazyka. Tyto otázky, podobně jako funkce, vybírají z množiny báze dat odpovídající podmnožinu.

Dotazovací jazyk SQL se skládá ze tří základních částí, které umožňují vytvářet následné konstrukce jazyka pro definici dat - DDL (Data Definition Language), který definuje nová data a jejich přesnou strukturu; jazyka pro manipulaci s daty - DML (Data Manipulation Language), který umožňuje tvořit dotazy a příkazy nad daty v databázi, tj. i vkládání, aktualizování a rušení záznamů v databázi; a jazyka pro definici přístupových práv k datům – DCL (Data Control Language), který řídí přístupová práva uživatelů (Vostrovský, 2004).

3.2.3. Datový model

Datový model je utvořen z entit, atributů, domén a vztahů.

- Entita - jedná se o abstrakci, která popisuje typ objektů. Je tvořena jménem a množinou atributů.
- Atribut - množina atributů dané entity zaznamenává skutečnosti - údaje o entitě.
- Doména - doména atributu vytváří množinu hodnot, která je reprezentovatelná v počítači, řetězce, čísla, booleovské hodnoty a další.
- Vztahy - u binárního[2] E-R modelu rozlišujeme vztahy s kardinalitou 1:1 a 1:N.

(Riordan, 2000; Pokorný, 2007)

3.2.4. Relační datový model

V roce 1969 byla poprvé uvedena relační databáze Dr. Edgarem F. Coddem z laboratoře IBM. Relační model byl výsledkem snahy nalézt nové způsoby spravování velkého množství dat. Dr. E. F. Codd se snažil pomocí matematického aparátu vyřešit

[2] existují i vztahy s kardinalitou M:N, ty je možno dekompozitovat na dva vztahy s kardinalitou N:1.

problémy s redundancí a slabou integritou dat. Relační model byl veřejně představen v článku *A relational model of data for large shared data banks* časopisu Communications of ACM v roce 1970.

V článku bylo popsáno:

- Oddělení dat od implementace.
- Symetrický přístup k datům, nejsou rozhodující mechanizmy přístupu k datům.
- Relační kalkul a algebra.
- Omezení nadbytečnosti dat je umožněno normalizací relace a vhodnou datovou strukturou.

(Pokorný, 1994)

Relační model je vystaven na matematické teorii množin a logických predikátech prvního řádu. Pojem „relace", vztah, je právě součástí teorie množin (Havrlant, 2011; Codd, 1970)

Databázová relace a relace v matematice se odlišuje pomocnou strukturou (tzv. schéma relace), která je tvořena jménem relace, atributů a domén. A také se odlišují prvky domén, které jsou tvořeny atomickými hodnotami a tvoří komponenty prvků relace. Vytváří 1. normální formu relací.

Kromě relačního modelu také existují hierarchické, síťové a objektově orientované modely dat. Tyto modely dat nejsou předmětem této práce.

Relační algebra

V relacích se s daty pracuje pomocí operací relační algebry: kartézský součin, sjednocení, rozdíl, selekce a projekce.

- Selekce - použitím operace selekce je vytvořena relace, obsahující jen ty záznamy z původní relace, které splňují podmínku stanovenou uživatelem.
- Projekce - operace relační algebry projekce umožňuje zredukování počtu atributů[3]. Z původní relace o n atributech se vytvoří relace o p atributech. Za předpokladu, že $p < n$.

[3] V relační algebře je atribut sloupec relace.

- Sjednocení - spojením dvou relací vytvoříme relaci, která obsahuje všechny možné kombinace.

(Vostrovský, 2004)

Datová normalizace

Datová normalizace umožňuje navrhnout relační databázi, reprezentující jakékoli typy relací, odstraňuje duplicity záznamů (např. množství vztahů vyplývajících z duplicity záznamů komplikuje aktualizaci) a dovoluje snadnější způsob přidání nových typů dat. Datová normalizace je reprezentována šesti úrovněmi. V praktickém použití se normalizace provádí do 3. normální formy, respektive BCNF (Boyce – Coddova normální forma). S vyšší úrovní roste i zatížení databázového systému, a tím klesá výkonnost.

- První normální forma – 1NF vylučuje existenci závislosti více hodnot na jedné klíčové hodnotě. Relace nesmí obsahovat násobná data.
- Druhá normální forma – 2NF stanovuje funkční závislost všech neklíčových dat relace na celém primárním klíči.
- Třetí normální forma - 3NF

  > *Schéma relace R(A) je ve třetí normální formě, jestliže žádný její neklíčový atribut není tranzitivně*[4] *závislý na žádném klíči R.* (Pokorný , 2007, s. 21)

  3NF je velice důležitá z důvodu aktualizací. Docházelo by k mnohačetné změně ve všech n-ticích, kde se daný atribut mění.
- Boyce - Coddova normální forma – BCNF vychází ze stejné definice jako 3NF a zpřísňuje ji i na vztah mezi hodnotami složeného primárního klíče (Pokorný, 2007).

3.3. Databázový server MySQL

3.3.1. Historie MySQL

Michael Widenius, David Axmark a Allan Larsson v roce 1995 založili ve Švédsku společnost MySQL AB. Od roku 2000 se stal projekt MySQL dostupný i pod licencí GPL. První verze byly dostupné pod operačním systémem UNIX respektive LINUX. Verze pod

[4] X -> Y -> C, C závisí tranzitivně na X.

MS Windows byla uvedena v roce 1998. V roce 2008 MySQL přešlo pod Sun Microsystems a zakladatelé M. Widenius a D. Axmark krátce poté vedení opustili. O rok později přešel Sun Microsystems pod vedení Oracle.

Významnými milníky ve vývoji MySQL bylo přidání podpory *union* (metoda spojení více výsledků dotazů) ve verzi MySQL 4.0 (2003) a o rok později ve verzi 4.1 přidání podpory R-tree indexů, podpora poddotazů, podpora Unicode (UTF8 a UCS2). Ve verzi MySQL 5.0 (2005) došlo k rozšíření o Úložné procedury, triggery, pohledy a přidání databáze *information_schema*. Výrazné změny přinesla verze 5.1, mezi které patří plná podpora triggerů, doplnění o plánování úloh, log operací MySQL tabulek, Clustery, pluginy[5] a možnost replikace.

3.3.2. MySQL a typy verzí

Databáze MySQL je rozdělena do čtyř typů instalace. Toto rozčlenění je dáno možností provozu MySQL pod licencí GPL (General Public License).

- MySQL Classic - standardní typ úložiště.
- MySQL Standart - standardní typ úložiště a InnoDB.
- MySQL Pro - licencovaná verze MySQL Standard.
- MySQL Max - obsahuje prvky, které nejsou ještě k dispozici v daných revizích.

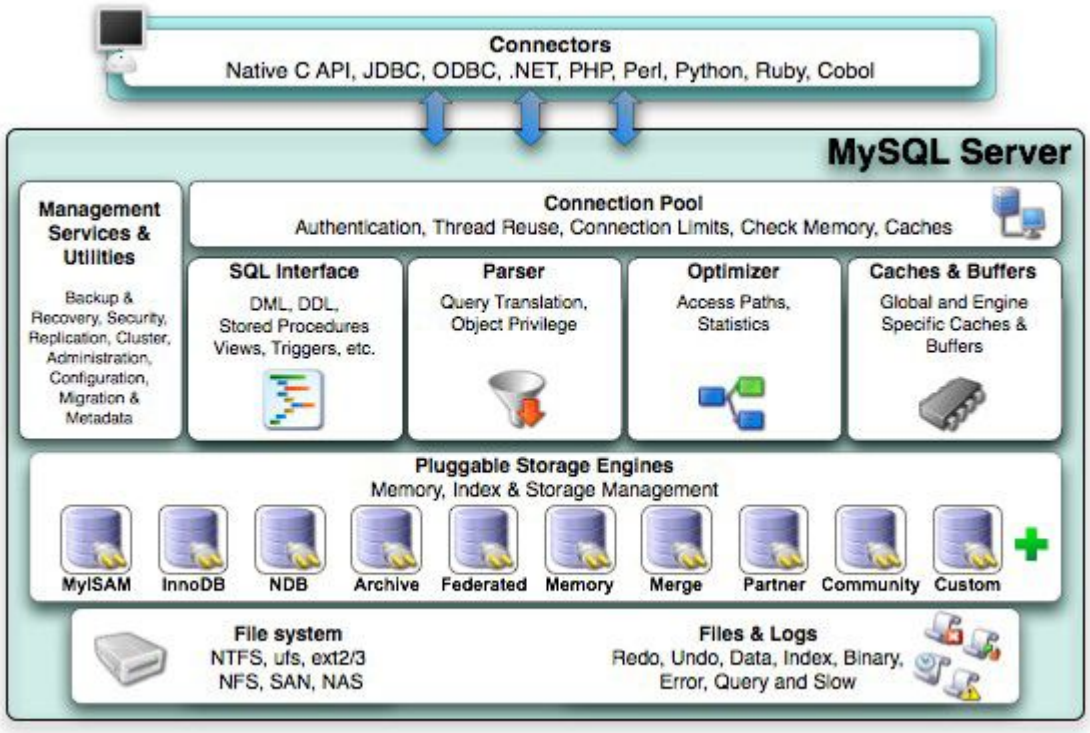
Používání MySQL Pro nevyžaduje zveřejňování zdrojového kódu, který s databází MySQL pracuje. Ostatní verze jsou distribuovány pod licencí GPL (Rosebrock, 2005).

3.3.3. Architektura serveru MySQL

Logická architektura serveru MySQL je znázorněna na obrázku č. 1: Schéma MySQL Serveru. První vrstva je tvořena službami, zajišťuje komunikaci klient / server, související s autentizací, bezpečností a vznikem vláken. Ve druhé vrstvě je prováděn parsing SQL, tj. rozebrání dotazu a vytvoření stromové struktury, tzv. parse-tree. Poté dochází k optimalizaci, určení pořadí přístupu k jednotlivým tabulkám, či určení potřebných indexů. V této vrstvě je také oblast cache dotazů, která obsahuje již dříve provedené rozbory dotazů pro použité

[5] Od verze MySQL 5.1 můžou být i úložné enginy reprezentovány jako pluginy. To umožňuje např. vytvořit rozhraní do jiného programu (Schwartz, 2009), či rozšíření zabezpečení správy přístupů (Cassar, 2011).

příkazy SELECT. Pokud se přijatý dotaz už vyskytuje v cache dotazů, neprovádí se parsing, optimalizace a ani vykonávání dotazu, ale je přímo vrácena výsledná sada pro daný dotaz. Ve třetí vrstvě jsou obsaženy úložné enginy. Úložné enginy komunikují se serverem prostřednictvím API jednotlivých úložných enginů a tím dochází ke sjednocení práce s různými typy úložných enginů ze strany serveru (Valenta, 2009; Schwartz, 2009).



Obrázek č. 1: Schéma MySQL Serveru (Valenta, 2009)

3.3.4. Druhy úložišť

MySQL obsahuje tři základní typy úložišť: MyISAM, InnoDB a MEMORY (do verze MySQL 5.0 označované HEAP). Dalšími úložnými enginy jsou MERGE, CSV, BDB (BarkeleyDB), ARCHIVE, NDB a FEDERATED. Úložný engin MyISAM je použit vždy jako výchozí (Schneider, 2006). Toto nastavení je možné změnit v konfiguračním souboru *my.cnf* v parametru *default-storage-engine*.

MyISAM

MyISAM je následovníkem formátu ISAM (Indexed Sequential Access Method) vyvinutém firmou IBM. Tabulka je tvořena primárním souborem a minimálně jedním souborem

indexů. Primární soubor obsahuje záznamy pevné délky uložené sekvenčně. Soubor indexu umožňuje vyhledávání záznamů podle zvolených atributů.

Tabulka MyISAM je reprezentována ve dvou souborech: *.MYD* (tvoří datový soubor) a *.MYI* (tvoří indexový soubor). Při vytváření tabulek server MySQL zvolí podle použitých datových typů druh statický, dynamický nebo komprimovaný.

Statický typ

Je použit pro data, obsahující pevně stanovenou velikost pro všechny sloupce tabulky. Přístup k takto uloženým datům je rychlý a vhodný pro data, která často upravujeme. V případě poškození souborů, ve kterých je tabulka uložena, je možné záznamy obnovit.

Dynamický typ

Je zvolen, pokud je v definici tabulky použit datový typ VARCHAR, TEXT, nebo BLOB. Paměťové nároky se sníží oproti statickému typu MyISAM. Záznamy využijí jen tolik paměti, kolik je jejich velikost. Negativem je problém při pozdější úpravě (aktualizaci) dat. Může dojít ke změně jejich velikosti oproti původní. A následně ke změně umístění těchto dat v databázovém souboru. Vznikají prázdná místa. Vznik nesouvislých bloků a fragmentace dat vede k prodlužování přístupové doby k tabulce. Je nutné pravidelně použít příkaz *OPTIMIZE TABLE*. Jinou možností je spuštění optimalizačního programu *myisamchk*.

Komprimovaný typ

MyISAM tabulky lze programem *myisamchk* komprimovat. Před každým čtením musí dojít k dekomprimování dat. Komprimované tabulky MyISAM jsou jen pro čtení, jejich obsah nelze měnit (Schwartz, 2009).

InnoDB

Typ tabulek InnoDB je integrován do MySQL od verze 3.23.24. Ovladač je vytvořen a podporován společností Innobase. Tento typ podporuje pomocné funkce pro lepší správu.

Operace nad tabulkami InnoDB se mohou spustit jako transakce. To umožňuje zvýšení bezpečnosti. Operace se může skládat z logicky souvisejících příkazů. Ty se buď provedou všechny, anebo žádný. Podle standardu ANSI-SQL/92 se transakce spouští v izolačních úrovních *READ UNCOMITTED, READ COMMITED, REPEATABLE READ, SERIALIZABLE.*

Při transakci dochází k zamykání na úrovni záznamů, pro ostatní uživatele nedochází k omezování přístupu.

Referenční integrita je spravována ovladačem InooDB. Při použití příkazů *DELETE, INSERT, UPDATE* nedojde k situaci, kdy je odkazováno na neexistující záznam jiné tabulky (Gilmore, 2007; Schwartz, 2009).

Memory (Heap)

Tabulky vytvořené s volbou úložiště MEMORY jsou uloženy v paměti RAM, jsou dostupné jen během činnosti serveru MySQL. Indexy jsou typu hash nebo B-tree. Tento typ úložiště je vhodný k rychlé a krátkodobé komunikaci mezi procesy. Srovnání rychlosti v praktickém měření: „ *[…] vytvořili jsme tři identické kopie tabulky message_buffer, které se liší pouze druhem úložiště. Vybrali jsme možnosti úložiště MYISAM, INNODB a MEMORY. Po vytvoření tabulek jsme do všech načetli velké bloky náhodných dat. Tyto operace byly u tabulek MEMORY v průměru o 33% rychlejší než u MYISAM a o 50% rychlejší než u INNODB.*“ (Schneider, 2006, s. 85).

Merge

Tento typ je určen pro rozložení zátěže jedné tabulky MYISAM na více shodných podtabulek (Schneider, 2006).

CSV

CSV (Comma Separated Values) je typ textových souborů, ve kterém jsou data oddělena čárkami, tvoří tabulku. Není podporována tvorba indexů. Kopírování souborů CSV je možné i za běhu serveru. Tento formát se nejčastěji používá pro výměnu dat, například export z tabulkového procesoru (Schneider, 2006).

BDB (BarkeleyDB)

BDB je transakční úložný engin starší než InnoDB. Umožňuje zamykání na úrovni stránky a podporuje řadu možností konfigurace (Schneider, 2006).

Archive

Úložný engine Archive je optimalizován pro potřeby komprimovaného uložiště s velmi rychlým vkládáním (Schwartz, 2009). Umožňuje pouze dotazy INSERT a SELECT.

Každý řádek, který je uložen, je navíc komprimován *zlib*[6]. Tento typ je vhodný například pro uchovávání logů (Schneider, 2006).

NDB Cluster

Všechna data jsou uchovávána v paměti. Na disk se ukládají jen logy. Je uzpůsoben k vyhledávání podle primárního klíče. NDB Cluster je svojí architekturou podobný poli RAID. Databáze je tvořena z datových uzlů, správních uzlů a SQL uzlů. Datový uzel, který je duplikován, udržuje jen část dat celého clusteru. Správní uzly slouží k centralizované konfiguraci a pro monitorování, řízení uzlů (Schneider, 2006).

Tato architektura je zcela odlišná od clastrů Oraclu.

Federated

V typu Federated nejsou data uložena lokálně, ale jsou odkazována na celé tabulky, uložené na jiném MySQL serveru (Schneider, 2006).

Blackhole

Nejedná se o ukládací mechanismus. Každý příkaz INSERT je zahozen. Všechny dotazy jsou ale logovány. Využití je při replikaci a auditu.

Falcon

Falcon je navržen pro servery s vícejádrovými 64bitovými procesory a s rozsáhlou operační pamětí. S použitím MVCC (Multiversion Concurrency Control) se snaží udržet všechny běžící transakce v paměti. To vede k zrychlení zotavujících operací a rychlému rušení transakcí. Dostupný ve verzi MySQL 6.0 (Schwartz, 2009).

solidDB

SolidDB je transakční engine využívající architektury MVCC. Obsahuje podporu cizích klíčů, pesimistické a optimistické řízení souběžnosti a zálohování online.

PBXT

Primabase XT je transakční engine používající MVCC a podporuje omezení cizích klíčů. Jeho architektura je charakteristická tím, že nepoužívá zápis nejdřív do logů, čímž sni-

[6] zlib – otevřená softwarová knihovna pro kompresi dat, vychází z kompresního programu gzip (Roelofs, 2012).

žuje režijní náklady potvrzení transakce. Se snížením času při potvrzování transakcí dochází k vysoké zapisovací souběžnosti. *„[...] různými testy už bylo prokázáno, že při jistých operacích může být rychlejší než engine InnoDB*.“ (Schwartz, 2009).

Maria

Engine Maria by měl v budoucnu nahradit engine MyISAM, ten je využíván jako výchozí úložný engine pro MySQL. Databázovým serverem je použit pro tabulky oprávnění a dočasné tabulky. Měl by povolovat volbu mezi transakčním a netransakčním úložištěm pro každou tabulku. V netransakčním režimu je možnost zotavení po havárii či zamykání na úrovni řádků a MVCC.

3.3.5. Datové typy

V databázi musí mít každý sloupec každé tabulky přesně definovaný datový typ. Data uložená v databázi MySQL tedy roztřiďujeme na následující datové typy.

Celá čísla

Datový typ INT je tvořen následujícími typy:

- TINYINT - 8bitové celé číslo, rozsah hodnot -128 až 127
- SMALLINT - 16bitové celé číslo, rozsah hodnot -32 768 až 32 767
- MEDIUMINT - 24bitové celé číslo
- INT - 32bitové celé číslo
- BIGINT - 64bitové celé číslo

Nastavením atributu *UNSIGNED* je omezen rozsah na nezáporná celá čísla. Atributem *AUTO_INCREMENT* je novému záznamu udělena hodnota o jednu větší než hodnota z předešlého záznamu. S tímto atributem se pojí následující specifika: každá tabulka může obsahovat nejvýše jeden atribut *AUTO_INCREMENT*, vyskytuje se společně s atributem *NOT NULL* a *PRIMARY KEY*[7] či *UNIQUE*.

Binární data jsou typu *BOOL* a jsou reprezentována *TINYINT*. Datový typ *BIT* je od verze MySQL 5.0.3 64bitový.

[7] PRIMARY KEY – primární klíč (index) jedinečným způsobem určuje každý záznam tabulky.

Desetinná čísla

Datové typy FLOAT(m, d) a DOUBLE(m, d) dodržují standardy IEEE[8]. Přesnost hodnot lze nastavit parametry *m* a *d*, *m* nastavuje celkový počet číslic a *d* reprezentuje počet číslic za desetinnou tečkou[9], je-li vstup delší, dojde k zaokrouhlení.

- FLOAT - 4 bajty, 8místná přesnost
- DOUBLE - 8 bajtů, 16místná přesnost, (REAL je synonymem pro d.t. DOUBLE)

Desetinná čísla s pevnou řádovou tečkou

Datový typ DECIMAL(p, s) reprezentuje desetinná čísla s pevnou řádovou tečkou. Tento datový typ se používá v případech vyžadujících vyšší přesnost než u datových typů FLOAT a DOUBLE, která jsou ovlivněna zaokrouhlovacími chybami. Čísla jsou uložena jako textové řetězce. Z toho plyne vyšší paměťová náročnost a nízký rozsah přípustných hodnot. Parametr *p* určuje celkový počet číslic a parametr *s* počet číslic za desetinnou tečkou. Jedna číslice je rezervována pro znaménkový znak (mínus). Záporné hodnoty jsou tedy vždy o jeden řád menší než kladné hodnoty. Příkladem je *DECIMAL(6, 3)* s rozsahem -999.999 až 9999.999. Synonymem pro tento datový typ je NUMERIC a DEC.

Datum a čas

- DATE - tvar data ‚2010-11-20', 3 bajty
- TIME - tvar času ‚23:59:59', 3 bajty
- DATETIME - tvar kombinace data a času ‚2010-11-20 23:59:59', 8 bajtů
- YEAR - rozsah roků 1900 – 2155, 1 bajt

Verze MySQL 5.0.2 umožňuje hlubší kontrolu správnosti data nastavením systémové proměnné *sql_mode*. Lze povolit vkládání i nekorektních dat, lze zakázat vkládání data 0000-00-00 (defaultně povoleno) nebo zakázat 0 pro měsíce a dny (též defaultně povoleno).

Datový typ TIMESTAMP slouží k automatickému ukládání časových údajů databází MySQL při jakékoli změně daného záznamu.

---

[8] IEEE – Institute of Electrical and Electronics Engineers (Institut pro elektrotechnické a elektronické inženýrství).

[9] Desetinná tečka – MySQL pracuje s čísly v mezinárodním formátu, nikoliv s desetinnou čárkou.

Textové řetězce

- CHAR(n) - text. řetězec pevné délky, až 255 znaků
- VARCHAR(n) - text. řetězec proměnné délky, až 255 znaků[10]
- TINYTEXT - text. řetězec proměnné délky, až 255 znaků

TEXT, MEDIUMTEXT a LONGTEX jsou textové řetězce proměnné délky až 2^{16} -1 znaků, 2^{24}-1 znaků a 2^{32}-1 znaků. Datový typ CHAR vyžaduje vždy pevnou délku stanoveného řetězce, je-li řetězec kratší, je doplněn mezerami. Při načítání nejsou tyto mezery načteny.

Délka textového řetězce u datového typu VARCHAR(n) je striktně omezena velikostí *n*, oproti tomu datový typ TEXT je omezen pouze svojí maximální délkou, viz výše. Krokem ke standardu ANSI[11] je od verze MySQL 5.0.3 upravení datového typu VARCHAR, který správně interpretuje mezery na konci textového řetězce. U sloupců s textovými řetězci lze nastavit atribut *CHARACTER SET znaková_sada COLLATE porovnávání,* a tím specifikovat znakovou sadu s porovnáváním. S verzí MySQL 4.1 byla přidána podpora znakových sad UTF-8[12] a UCS-2[13]. Do této verze nebyla Unicode podporována. Aktuálně podporované znakové sady a porovnávání lze zjistit zadáním příkazu SQL *SHOW COLLATION*.

Binární data

Datový typ BIT(n) pro n bitů, maximálně však 64 bitů. TINYBLOB, BLOB[14], MEDIUMBLOB a LONGBLOB reprezentují binární data s proměnnou délkou maximálně 255, 2^{16}-1, 2^{24}-1 a 2^{32}-1 bajtů. Všechny tyto datové typy mají obdobné vlastnosti jako TEXT, binární data jsou ale ukládána v binárním kódu.

[10] Od verze MySQL 5.0.3 je maximální počet znaků n < 65 536.

[11] ANSI – American National Standards Institute – Americký národní standardizační institut.

[12] UTF-8 – převodní formát pro Unicode, umožňuje zpětnou kompatibilitu se znaky ACSII (do znaku 127 včetně) a dále umožňuje zapsat 2^{31} různých znaků všech národních abeced. Definováno v ISO 10646-1:2000 Annex D (Bříza, 2012).

[13] UCS-2 – univerzální znaková sada Unicode/UCS, která neobsahuje zpětnou kompatibilitu s ASCII.

[14] BLOB – Binary large obect, velký binární objekt.

Datový typ ENUM, SET a pro GIS[15]

Tyto datové typy se vyskytují v databázi MySQL a jsou využity v nastavení databáze. Datový typ ENUM, výčet, pracuje s textovými řetězci označenými čísly v maximálním počtu 65 535. Datový typ SET, množina, má obdobný význam, je však omezen na maximálně 64 textových řetězců a jejich kombinací. Vnitřně jsou databází reprezentovány jako čísla, která jsou přiřazena textovým řetězcům při založení tabulky se sloupcem datového typu ENUM, nebo SET. Příkladem využití těchto datových typů je databáze *mysql* tabulka *user* a zde nastavení privilegií *enum('N', 'Y')*, nebo v tabulce *columns_priv* a sloupci *Column_priv set('Select', 'Insert', 'Update', 'References')*.

Datové typy pro geometrická data jsou dostupné od verze MySQL 4.1 a vycházejí ze standardu OpenGIS sdružení OGC[16]. Dělí se na geometrické typy ukládající přesně jeden objekt (GEOMETRY, POINT, LINESTRING, POLYGON) a na datové typy ukládající více objektů (GEOMETRYCOLLECTION, MULTIPOINT, MULTINESTRING, MULTIPOLY-GON) (Kofler, 2007). Možnosti práce s GIS jsou velice rozsáhlé a nejsou předmětem této práce.

Srovnání s db ORACLE a Microsoft SQL Server 2008

Databáze ORACLE obsahuje pouze jeden datový typ NUMBER(p, s) pro čísla celá či reálná. Typ čísla s pevnou řádovou tečkou (Fixed point) se nastavuje proměnnou *p*, která udává přesnost (precision) a proměnnou *s*, udávající měřítko (scale) v rozsahu od <-84, 127>. Čísla s plovoucí desetinnou čárkou mohou být s 38 platnými čísly v rozsahu 1.0×10^{-130} až $9.9 \ldots 9 \times 10^{125}$. Obecná podpora SQL dle standardu ANSI je prováděna db ORACLE mapováním na vlastní interní datové typy (Loney, 2010).

SQL Server 2008 od Microsoftu používá zvláštní datové typy pro práci s hierarchickými daty *hierarchyid*[17].

[15] GIS – Geografické informační systémy.

[16] OGC – Open Geospatial Consortium je společenství obchodních firem, univerzit a úřadů (Kofler, 2007).

[17] hierarchyid – nejedná se nativní datový typ, ale o systémem definovaný uživatelský typ (UDT) (Walters, 2009).

3.3.6. Úložné procedury, triggery a události

Úložné procedury, triggery a události jsou součástí vrstvy, vytvářející z databáze více, než jen prosté úložiště dat. Úložné procedury a triggery jsou dostupné od verze MySQL 5.0. Od verze MySQL 5.1 byla vylepšena podpora triggerů a přidána možnost periodického spouštění uloženého kódu serverem, tzv. events (události). Úložné procedury, resp. funkce SQL, triggery a události jsou spouštěny serverem MySQL. Úložné procedury a funkce přijímají parametry a následně odesílají výsledek. Triggery a události pouze vykonávají předem uložený kód. Syntaxe úložných procedur vychází ze standardu SQL 2003 a je i kompatibilní s úložnými procedurami IBM DB/2. U systémů Oracle a Microsoft SQL Server není standard dodržen a úložné procedury nejsou pro tyto systémy přenositelné. Úložné procedury, funkce jsou interně uloženy v tabulce *mysql.proc*. Sloupce tabulky *mysql.proc* evidují zejména údaje o názvu databáze (nad kterou se vykonávají), jméno, typ procedury (procedura nebo funkce), samotný kód a další. Periodicky se opakující události (events) jsou interně uloženy v tabulce *mysql.event*. Sloupce této tabulky udržují zejména údaje o názvu databáze, nad kterou se provádějí, jméno, samotný kód, informaci o periodickém spuštění (ročně, čtvrtletně, měsíčně …) a času vykonání. Interně jsou poté spouštěny plánovačem, který ale není defaultně nastaven. Plánovač se spustí přidáním příkazu `event_scheduler = ON` do souboru *my.*cnf (Schwartz, 2009; Kofler, 2007).

Triggery provedou uložený kód při zavolání příkazů INSERT, UPDATE a DELETE. Kód je proveden před či po volacím příkazu a je uložen v prostém textovém souboru *tabulka.TRG* v adresáři databáze. Omezení práce MySQL s triggery je, že pro každou událost (INSERT, UPDATE a DELETE) může být uložen pouze jeden trigger, který je vždy prováděn na úrovni řádků. Při použití triggeru nad tabulkami MyISAM nelze zajistit atomicitu těchto tabulek. I přesto, že dojde k chybě spouštěcího příkazu volajícího trigger, dojde k vykonání uloženého kódu, a tím může být narušena atomicita. Oproti tomu při provádění triggeru nad tabulkami InnoDB se provádí vše během jedné transakce. Tím je zajištěna atomicita spouštěcího příkazu a vykonaného triggeru. (Schwartz, 2009)

3.4. Zabezpečení

3.4.1. Obecné aspekty zabezpečení DBMS

Problematiku zabezpečení lze rozdělit podle odlišného přístupu k datům a k databázi. Prvním úhlem pohledu je administrátor dat (správa dat) a druhým je administrátor databáze (správa databáze). Hlavní rozdíl uvedených činností je v odbornosti. Práce administrátora dat je spíše podobná manažerské činnosti – jedná se například o správu a kontrolu dat celé organizace, aktuálnost standardů a všeobecné postupy, procedury a logické návrhy databáze. Zatímco náplň práce administrátora databáze je převážně odborně technicky orientovaná – je vyžadována realizace fyzického databázového systému - DBMS, implementace návrhu databáze a zabezpečení, ověřování integrity dat a sledování výkonnosti databázového systému (s tím související případná reorganizace dat). (Conolly, 2009, s. 289)

Ve víceuživatelském prostředí je podle T. Connolla kladen důraz na autorizaci, autentizaci, přístupová práva, pohledy, zálohování, žurnálování, časové okno zálohování, zašifrování a diskové pole RAID[18]. Ze strany zabezpečení sítě se jedná o samotné zabezpečení sítě, firewall a demilitarizovanou zónu. Autorizaci, autentizaci a přístupovým právům je věnována samostatná kapitola 3.5 Správa přístupů v MySQL.

Pohledy umožňují pružně přizpůsobovat bezpečnost. Uživatel přistupuje k části databáze, která je definována sloupci a řádky, složenými i z více než jedné tabulky. Pohledy rozšiřují možnosti využívání přístupových práv. Například přístupové právo k danému pohledu dovoluje získat pohledem daná data z jinak pro uživatele nepřístupných podkladových tabulek.

Důležitou částí komplexního zabezpečení databáze je její zálohování a s tím související žurnálování. Jedná se o proces pravidelného kopírování celé databáze a log souborů, tzv. žurnálů, zaznamenávajících aktualizace záznamů v databázi. Kombinace záloh a záznamů v log souboru umožňuje obnovení databáze do posledního konzistentního stavu.

Pro zabezpečení velice citlivých údajů v databázích, jako jsou čísla sociálního pojištění, kreditních karet, bankovních účtů, hesel atd. je využito šifrovacích algoritmů. Příkladem je americký zákon HIPAA[19] z roku 1996, část zákona klade přímo požadavky zabezpečení na

[18] RAID - Redundant Array of Inexpensive/Independent Disks, zabezpečení dat na pevných discích proti selhání.

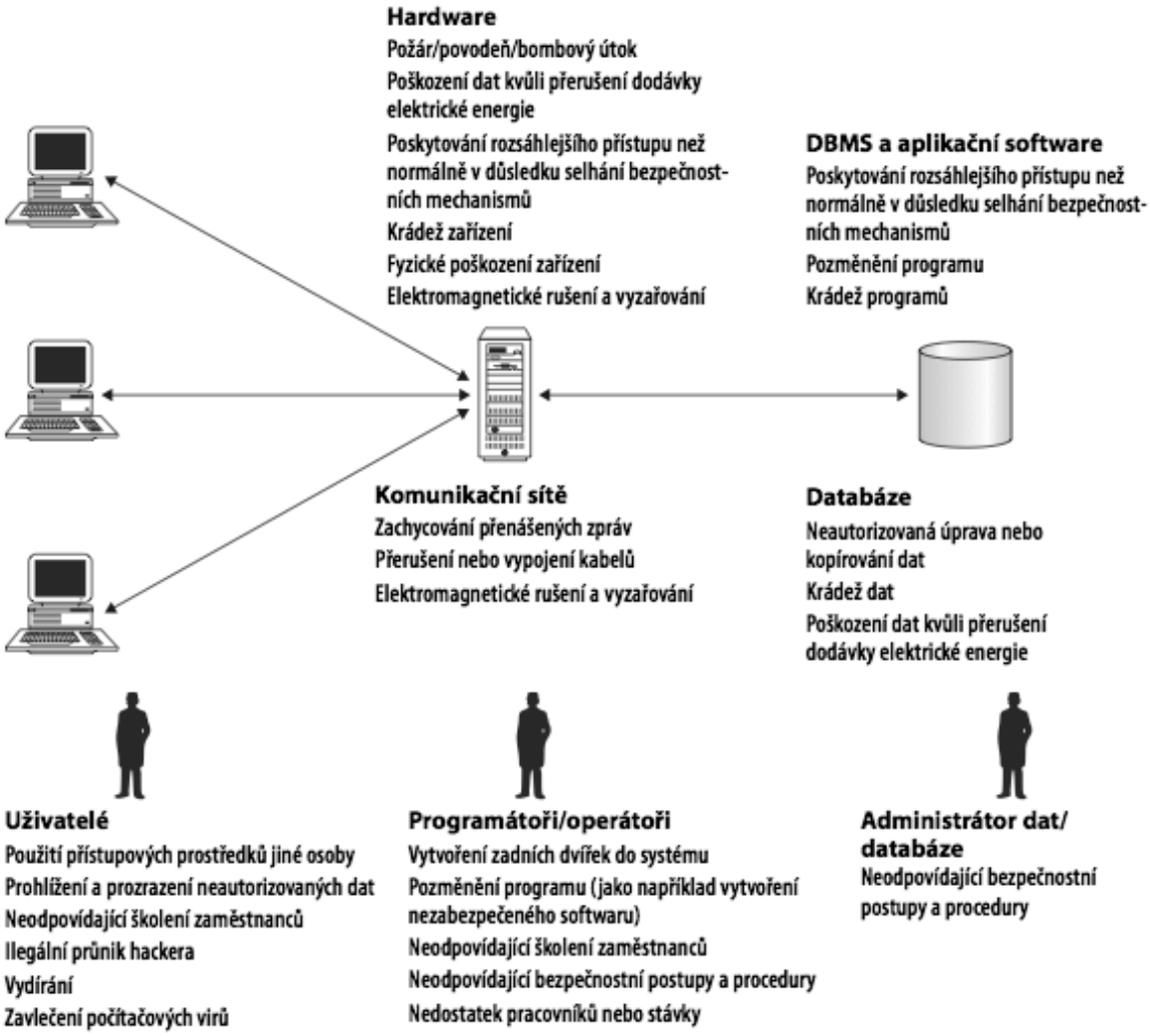
[19] HIPAA – Health Insurance Portability and Accountability Act, zákon ministerstva zdravotnictví USA.

úrovni správy databáze zdravotní péče a pojištění. Data musí být zabezpečena uvnitř databáze a i během přenosů. A s tím související standardizace dat pro přenos mezi organizacemi. U tohoto způsobu jsou chráněná data zašifrována a uložena. Při každém přístupu DBMS k datům dochází k dešifrování a tím i k poklesu výkonnosti serveru.

Způsoby zabezpečení přenosu dat nezabezpečenou sítí využívají šifrování symetrického a asymetrického klíče. Symetrický, stejný klíč pro zašifrování a dešifrování dat je ve srovnání s algoritmy asymetrického klíče rychlejší. Asymetrický klíč využívá rozdílné klíče pro zašifrování a dešifrování. Například veřejný klíč a tajný klíč. Využití obou metod v praxi je s protokolem SSL (Secure Socket Layer), zabezpečujícím přenos mezi webovým serverem a uživatelem (webovým prohlížečem) (Conolly, 2009). Zabezpečení protokolem SSL je věnována kapitola 4.1.4 Zabezpečené přístupy k MySQL.

Demilitarizovaná zóna je použita v tzv. třívrstvém databázovém systému[20], kdy jsou jednotlivé vrstvy odděleny firewallem. Příkladem je vnější síť internet, oddělená *vnějším firewallem* od webového serveru a ten je oddělen *vnitřním firewallem* od databázového serveru (Conolly, 2009). V prostoru mezi firewally vznikne demilitarizovaná zóna.

[20] Třívrstvá architektura databázového systému – odděluje klienty, na straně serveru aplikační služby a datové služby. Umožňuje také vhodnější rozložení zátěže.



Obrázek č. 2: Shrnutí potencionálních ohrožení počítačových systémů (Conolly, 2009, s. 296)

3.4.2. Zabezpečení ve vztahu k MySQL

Na zabezpečení ve vztahu k databázi MySQL lze nahlížet ze tří pohledů. Zabezpečení vlastního databázového stroje po stránce fyzické, po stránce programového vybavení a také zabezpečení samotných aplikací přistupujících k databázi. Pro programové vybavení se jedná zejména o operační systém a na něm provozované služby, např. operační systém Linux, servery Apache a MySQL, PHP. Druhým pohledem na problematiku zabezpečení je správa přístupů a oprávnění pro práci s konkrétní databází. A třetím pohledem je zabezpečení na straně programů přistupujících k databázi.

3.4.3. Zabezpečení „databázového stroje“

Bezpečnostní riziko je na straně klienta (aplikace – redakční systém apod.) komunikujícího s databází. Je nutné zajistit zabezpečení přístupových údajů, sloužících pro navázání spojení s databázovým serverem. Příkladem je zabezpečení souborů skriptovacího jazyka PHP, které ukládají přihlašovací údaje pro navázání spojení s databázovým serverem, proti neoprávněnému přečtení návštěvníkem v podobě prostého textu. Zdrojový kód PHP je interpretován webovým serverem, měl by tedy vždy být viděn jen HTML dokument. Může ale dojít k chybám na straně serveru, které způsobí zobrazení souborů PHP bez jejich správné interpretace. Nejčastěji k tomuto může dojít při nesprávné aktualizaci webového serveru, při nevhodném nastavení anonymního přístupu přes FTP k daným složkám, nebo při vytváření záložních kopií úpravou přípony souborů, např. *soubor.php.bak* (Kofler, 2007). Bezpečná konfigurace na straně serveru je uvedena v samostatné kapitole v praktické části práce.

V zabezpečení prostřednictvím PHP 5 lze postoupit dále a využít objektových vlastností a celý kód, pracující přímo s databází, zapouzdřit do nové třídy.

Přihlašovací údaje, jméno a heslo, přesto zůstávají uloženy v prosté textové podobě.

3.5. Správa přístupů v MySQL

Zabezpečení přístupů k MySQL je dvoustupňové. Prvním stupněm je autentizace uživatele a druhým stupněm je ověření oprávnění pro uživatele.

3.5.1. Oprávnění

Pro udělení oprávnění nejprve MySQL autentizuje uživatele podle uživatelského jména, hesla a hostitele[21]. V další fázi dochází k autorizaci, tj. zjištění, jaké oprávnění je uživateli přiděleno (Schwartz, 2009). Oprávnění uživatele odpovídá spouštěnému dotazu (*SpouštěnýDotaz_priv* a nabývá hodnot Y / N, ano / ne). Např. pro udělení práva na odstranění záznamu v tabulce je nastavena hodnota oprávnění *Delete_priv = Y*.

Rozlišujeme dva druhy oprávnění, dělená podle sdružení s objekty. Objektem jsou databáze, tabulky, pohledy a triggery (Kofler, 2007).

[21] MySQL při autentizaci rozlišuje hostitele – umístění, které je specifikované názvem hostitele, IP adresou či zástupným symbolem.

Oprávnění specifická pro objekty

Oprávnění specifická pro objekty se udělují pro přístup ke zvoleným objektům. Udělují oprávnění pro získávání dat z databáze, tabulky či pohledu.

Globální oprávnění

Globální oprávnění dovolují provádět úpravy s databázovým serverem. Evidovány jsou v tabulce *mysql.user*. Nastavením příslušných oprávnění v daném sloupci (*Oprávnění_priv*) pro zvoleného uživatele, uložením hodnoty Y / N, ano / ne. Nastavení hodnoty N znamená odepření globálního oprávnění uživateli a to následně vede ke kontrole objektového oprávnění. Přidělování globálních oprávnění má vliv na bezpečnost celého serveru. Například oprávnění umožňující restart serveru – *Reload_priv* umožňuje spuštění příkazu *FLUSH*[22] (Kofler, 2007; Schwartz, 2009).

3.5.2. Kontrola oprávnění

Informace o udělených oprávněních MySQL ukládá v databázi *mysql* v tabulkách typu MyISAM.

Kontrola oprávnění je prováděna v pořadí naznačeném na obrázku č. 3: Kontrola oprávnění v MySQL. Je-li nalezeno povolení oprávnění, je proces kontrol ukončen a přechází se přímo na vykonání dotazu (Schwartz, 2009, s. 550).

Tabulka *user*

Určuje uživatele - uživatelský účet, jméno, hostitel a heslo. V této tabulce udělujeme globální oprávnění a od verze MySQL 4.0 i limity[23], nastavení SSL[24] šifrovaného spojení a nastavení identifikace X.509 (standard ověřování identity).

Sloupce *Host, User* a *Password* jsou citlivé na velikost písmen. Pouze v nastavení IP adresy nebo názvu počítače ve sloupci *Host* lze použít zástupný znak % a _. Procento zastupuje libovolný textový řetězec a podtržítko jeden jakýkoliv znak. Je-li prázdný záznam sloupce *User*, jedná se o anonymního uživatele. Ukládání hesla do sloupce *Password* se

[22] Příkazem *FLUSH* jsou všechny údaje, které ještě nebyly zapsány, uloženy do databáze. Pro spuštění tohoto příkazu je nutné oprávnění *RELOAD*. (Kofler, 2007, p. 681)

[23] Limity na počet připojení a provádění dotazů za jednu hodinu. Využití limitů v kontextu bezpečnosti je věnována kapitola 4.2.4 Omezení serveru limity připojení.

[24] SSL – secure socket layer, podrobněji viz. kapitola 4.1.4 Zabezpečené přístupy k MySQL.

provede funkcí PASSWORD()[25]. Prázdný záznam sloupce *Password* umožňuje přihlášení uživatele bez zadání hesla.

Tabulka *db*

Řádek je tvořen oprávněním uživatele k dané databázi.

Tabulka *host*

Řádek tabulky *host* tvoří oprávnění k dané databázi pro uživatele. Uživatel je připojen z daného hostitele. Tuto tabulku oprávnění nelze upravovat příkazy *GRANT* a *REVOKE*, lze ji upravovat pouze ručně. Položky tabulky *db* a *host* jsou sloučeny – marget.

Tabulka *tables_priv*

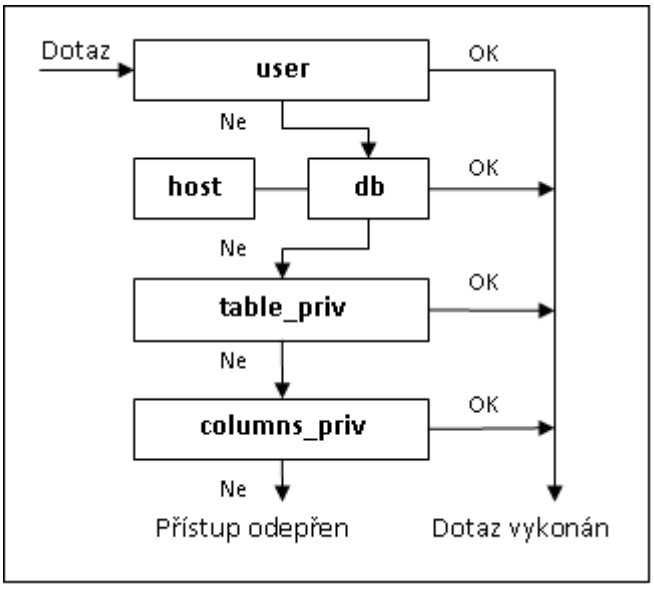
Řádek tabulky *tables_priv* specifikuje oprávnění k dané tabulce pro uživatele. Zahrnuje i oprávnění pro pohledy.

Tabulka *columns_priv*

Řádek tabulky *columns_priv* tvoří oprávnění uživatele k danému sloupci.

Tabulka *procs_priv*

Řádek tabulky *procs_priv* obsahuje oprávnění pro uživatele a úložné rutiny.



Obrázek č. 3: Kontrola oprávnění v MySQL (Schwartz, 2009, s. 550)

[25] SQL funkce PASSWORD() zašifruje heslo do Hash (41 znaků).

3.6. Způsoby zabezpečení

3.6.1. Úložné rutiny

Úložné procedury umožňují zpřesnit kontrolu nad oprávněními. Příklad využití pro přenos finančních prostředků v bance: *„[...] procedura přenáší peníze uvnitř transakce a zaznamenává do logu celou operaci pro potřeby auditu. Aplikacím můžeme dovolit, aby volaly tuto uloženou proceduru, aniž byste jim museli udělit přístup k podkladovým tabulkám.*“ (Schwartz, 2009, s. 243).

Použití úložných procedur předpokládá, že nebudou vždy spouštěny uživatelem, který je vytvořil. Udělení oprávnění se řídí oprávněním uživatele, který úložnou proceduru vytvořil. Úložná procedura přistupuje k databázi s oprávněním uživatele, který proceduru vytvořil. Přidělením oprávnění k využití procedury jinému uživateli dojde o rozšíření oprávnění uživatele o přístupy k podkladovým tabulkám skrze úložnou proceduru. Bezpečnostním rizikem je možnost opomenutí udělení oprávnění uživatelům k úložným procedurám a tedy i skrze ně přístupu ke konkrétním datům vlastníka procedury (DuBois, 2009).

Bezpečnostním rizikem souvisejícím s použitím úložných rutin je jejich uložení společně s databází. „*Pokud například budete mít uvnitř uložené rutiny nějakou nestandartní kryptografickou funkci, neuchráníte svá data, pokud bude databáze napadena.*“ (Schwartz, 2009, s. 244).

3.6.2. Udělování oprávnění

Použití příkazů GRANT a REVOKE

Použitím příkazu *GRANT* můžeme udělovat oprávnění a příkazem *REVOKE* je odebírat. Po použití těchto příkazů se nemusí provádět příkaz *FLUSH PRIVILEGES*, jako tomu je při přímém přístupu k databázi *mysql* s využitím běžných příkazů *INSERT, UPDATE* a *DELETE*. Z důvodu zrychlení práce s databází *mysql* je kopie této databáze uchovávána v paměti RAM. Reálně je změna provedena právě příkazem *FLUSH PRIVILEGES* (Kofler, 2007).

```
GRANT oprávnění
 ON [database.]tabulka nebo database.nazevÚložnéProcedury
 TO uživatel@host [IDENTIFIED BY 'heslo']
```

```
[WITH GRANT OPTION]
```

Pro nastavení globálních oprávnění se použije ON *.*. Syntaxe udělení oprávnění pro všechny tabulky dané databáze je ON *databáze.**. Pro zadání oprávnění všem uživatelům z daného hostitele lze použít ‘’, prázdné jednoduché uvozovky. V části *host* lze použít zástupného symbolu %.

Uživatel s právem vytvářet „vlastní“ databáze

V případě většího množství uživatelů je administrativně náročné vytvářet zvlášť pro každého databázi. Každému uživateli je uděleno právo na vytváření databází. Název nových databází musí obsahovat prefix uživatele, *prefix_názevDatabáze*, nejčastěji se skládající ze jména uživatele. Uživatel bude moci pracovat pouze s databázemi začínajícími jeho specifickým prefixem.

```
INSERT INTO mysql.db (Host, Db, User, Select_priv, Insert_priv,
  Update_priv, Delete_priv, Create_priv, Drop_priv, Grant_priv,
  References_priv, Index_priv, Alter_priv, Create_tmp_table_priv,
  Lock_tables_priv, Create_view_priv, Show_view_priv)
VALUES ('localhost', 'uzivatel\\_%', 'uzivatel', 'Y', 'Y', 'Y',
  'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y')
```

Například provozovatel webhostingových služeb Active24.cz: *„[...] uvedený způsob, s prefixem databáze např. podle uživatele, není u sdíleného hostingu využíván. Používána je výhradně konfigurace 1:1, tedy jeden uživatel k jedné databázi s tím, že se databáze jmenuje stejně jako uživatel. Hlavním důvodem je evidence a přehlednost.*“ vedoucí technického oddělení Active24.cz (Hála, 2012). Oproti tomu společnost Český hosting: „*[...] pro vytváření uživatelů a databází používáme jmennou konvenci. Jeden uživatel může mít přístup i do více databází.*“ (Mach, 2012).

4. Vlastní práce

Problematika zabezpečení databáze je komplexní problém přesahující samotnou databázi. Lze ji rozdělit do následujících tří částí, kterým se musí věnovat při konfiguraci pozornost. Prvním aspektem je zabezpečení přístupu, připojení a autentifikace. Pokud nebude věnována dostatečná pozornost bezpečnosti klientů připojujících se k databázi a jejich autentifikaci, může být databáze zranitelná. Druhým aspektem je bezpečnost DBMS po stránce softwarové a hardwarové. Způsoby, jak předejít, nebo alespoň zmírnit následky selhání databáze pomocí zálohování a obnovení, redundance a jiných, tvoří architekturu HA[26] (vysoká dostupnost). Třetím a posledním aspektem je ochrana proti nežádoucím přístupům, uchování integrity, konzistence databáze a obecné využití DDL a DCL (DuBois, 2009; Valenta, 2012).

4.1. Zabezpečení přístupů k DB

4.1.1. Zásady udělování oprávnění

Pro zvýšení bezpečnosti je důležité zamezit přístupu bez hesla a uživatelského jména, tj. anonymního uživatele. MySQL oba tyto případy umožňuje. Schwartz (2009) doporučuje přidat do konfiguračního souboru *my.cnf* následující parametr:

```
[client]
password
```

Tímto parametrem je vždy uživatel vyzván k zadání hesla. Od verze MySQL 5.0 lze nastavením NO_AUTO_CREATE_USER omezit příkaz GRANT.

Účinným nástrojem pro odstranění všech položek z tabulek oprávnění, které obsahují *User* jako prázdný řetězec, tedy anonymního uživatele, je program *mysql_secure_installation*, který tuto úpravu provede (Schwartz, 2009).

[26] HA (High Availability) – vysoká dostupnost.

4.1.2. Zabezpečení přístupů k db z CMS Drupal

Databáze MySQL je v současné době velice často používána ve spojení s PHP a vytváří publikační (redakční) systém, zkráceně CMS[27] (Content Management System), příkladem jsou Drupal, WordPress a Joomla!. Poskytovatelé hostingových služeb, například Active24.cz a c4.cz, umožňují uživatelům instalaci těchto redakčních systémů velice jednoduchým způsobem. Při konfiguraci vlastního hostingového prostoru nabízejí automatickou instalaci jimi podporovaného redakčního systému, nebo mírně upraveného balíku. Například uvedený provozovatel c4.cz upravuje v instalačním balíku pro CMS Drupal[28] soubory *.htaccess, settings.php* a přidává podporu češtiny (Drupal 7, 2012; Instalace webu jedním kliknutím, 2012).

Instalace CMS Drupal spočívá ve čtyřech krocích. Stažení a rozbalení aktuální verze na server, vytvoření databáze (včetně nového uživatele), nakonfigurování souboru *settings.php* a spuštění instalačního scriptu. Při „Instalaci jedním kliknutím“ u provozovatele Active24.cz jsou všechny kroky provedeny automaticky a uživatel obdrží přihlašovací údaje k účtu admin CMS Drupal a přístupové údaje k vytvořené MySQL databázi e-mailem.

Přihlašovací údaje k databázi jsou uloženy v souboru *settings.php.* Tento konfigurační soubor je uložen v adresářové struktuře */sites/default/settings.php*.

4.1.3. Bezpečnostní rizika ve vztahu k databázi MySQL

Zranitelnost uvedené aplikace je v uložených přístupových údajích k databázi MySQL. Tyto údaje se vyskytují v prosté textové podobě v konfiguračním souboru (např. právě výše uvedený soubor */sites/default/settings.php*).

Vhodně zvolená oprávnění adresářů a konfigurace serveru umožňuje ochranu takto citlivých údajů proti zcizení.

---

[27] CMS – zpravidla webová aplikace umožňující správu obsahu a dokumentů, tvořená administrátorským a uživatelským rozhraním. Ve webovém prostředí např. umožňující tvorbu článků pomocí WYSIWYG editoru (z ang. *What you see is what you get,* „Co vidíš, to dostaneš“, umožňují tvorbu web. obsahu bez znalosti jazyka HTML), správa uživatelů a jejich práv či správa diskuzí.

[28] CMS Drupal je distribuován pod licencí GPL a umožnuje tyto úpravy. Vývoj CMS Drupalu byl zahájen D. Buytaertem v roce 1999. Aktuální verze Drupal 7.x v základní konfiguraci podporuje databáze MySQL 5.0.15 a vyšší (případně obdobnou databázi MariaDB[28] od verze 5.1.44), PostgreSQL 8.3 a SQLite 3.x (System requirements, 2012).

Konfigurace serveru Apache

Bezpečnostní riziko lze účinně snížit uložením přihlašovacích údajů do samostatného souboru v samostatné složce, správným nastavením oprávnění a vlastností serveru (na níže uvedeném příkladě serveru Apache). Přístup do této složky je zabezpečen nastavením souboru *.htaccess*. Jmenovaný soubor umožňuje při použití serveru Apache upravit konfiguraci serveru pro konkrétní adresář, ve kterém je soubor *.htaccess* uložen. Toto řešení kromě bezpečnosti snižuje i redundantnost zdrojového kódu a zlepšuje celkovou zprávu projektu. Přihlašovací údaje k databázi měníme pouze na jednom místě (Kofler, 2007).

4.1.4. Zabezpečené přístupy k MySQL SSL a SSH

Vzdálený přístup k serveru přináší bezpečnostní riziko. Zpravidla se jedná o přístup skrze veřejnou nezabezpečenou síť, umožňující komukoli odposlechnout celou komunikaci klienta se serverem. K zabezpečení této komunikace se používá celá řada kryptografických nástrojů.

Prostřednictvím SSL

Pro zabezpečený přenos dat skrze obecně nezabezpečenou síť Internet je využívána technologie SSL (Secure Socket Layer). Obecně pro zajištění bezpečnosti je nutné, aby se odesílatel a příjemce dobře znali. Tento předpoklad je v technologii SSL využit skrze tzv. Certifikační autority. Certifikační autorita je organizace vydávající bezpečnostní certifikáty. Certifikát slouží například k ověření pravosti elektronického podpisu odesílatele. Za autentičnost ručí Certifikační autorita, která daný certifikát vydala danému uživateli. Celosvětově známé certifikační agentury jsou RSA Security, VeriSign Inc. a v České republice PostSignum (Česká pošta).

SSL zpravidla není automaticky nastavené v každé distribuci MySQL serveru. Informace o podpoře dané instalace je uložena v systémové proměnné *have_openssl*. MySQL server musí být zkompilovaný s podporou SSL, není tedy možné dodatečně tuto podporu pouze zapnout. Je-li SSL podporováno, lze pro zabezpečení přístupů k MySQL příkazem GRANT povolit uživateli připojení jen přes SSL.

```
mysql> GRANT ... IDENTIFIED BY 'p4ssword' REQUIRE SSL;
```

(Schwartz, 2009, s. 572)

Tímto příkazem není vyžadováno ověření platnosti certifikátu. Změnou v příkazu za `REQUIRE X509`[29]`;` lze vyžadovat ověření proti certifikátům certifikačních autorit, které jsou pro server MySQL nastaveny. Možností omezení přístupu skrze SSL je více. Lze vyžadovat jakékoli certifikáty vydané požadovanou certifikační autoritou, nebo jen specifický certifikát vydaný pouze konkrétní certifikační autoritou. Tato varianta je uvedena v následujícím příkladu.

```
mysql> GRANT ... IDENTIFIED BY ‚pa4ssword‘
    -> REQUIRE SUBJECT "/C=US/ST=New York/L=Albany/0=Widgets
Inc./CN=client-ray.example.com/emailAddress=raymond@example.com"
   -> AND ISSUER "/C=US/ST=New+20York/L=Albany/0=Widgets
Inc./CN=cacert.example.com/emailAddress=admin@example.com";
```

(Schwartz, 2009, s. 573)

Část příkazu REQUIRE SUBJECT povoluje pouze konkrétní certifikát a REQUIRE ISSUER specifikuje certifikační autoritu. Vytvářené klientské certifikáty obsahují datum expirace, tedy datum, kdy skončí jejich platnost. Vhodně nastavená perioda obměny klientských certifikátů zvyšuje bezpečnost. Například podle Schwartze (2009) v prostředí rozsáhlé firmy s mnoha zaměstnanci, přistupujícími k databázi MySQL s certifikáty, lze bezpečnost zvýšit pravidelně přidělovanými certifikáty na jeden nebo dva měsíce. Neobnoví-li se platnost certifikátu, zaměstnanci nebo obchodnímu partnerovi dojde po uplynutí expirace k zamezení přístupu do databáze, i když je stále jeho účet platný (Schwartz, 2009; Barrett, 2003).

Prostřednictvím SSH

Zabezpečený přístup nazývaný „tunelování“ označuje zapouzdření komunikačního protokolu do šifrovaného SSH (Secure Shell) spojení. Podporováno je směrování služeb TCP/IP, X11[30] a tzv. agentů umožňujících klientům SSH použít soukromé klíče (Barrett, 2003, s. 45) .

[29] X509 – standard systémů s veřejným klíčem (Cooper, 2005).

[30] X11 – Zabezpečení protokolu pro X Window System.

Přístup skrze protokol SSH umožňuje využití veřejných kryptografických nástrojů pro zajištění autentizace účastníků, jejich soukromí a také pro kontrolu integrity celého síťového spojení.

4.1.5. Bezpečnost šifrování a heslo uživatele

MySQL obsahuje hashovací funkce vytvářející obraz, neboli hash, chráněného textového řetězce. Jedny ze starších a bezpečnostně rizikových funkcí jsou MD5(), OLD_PASSWORD(), PASSWORD(), SHA() a SHA1(). Tyto funkce obsahují algoritmy, které nejsou dostatečně odolné vůči útokům. Kryptografická funkce přidaná ve verzi MySQL 5.5 SHA2() využívající algoritmy SHA-224, SHA-256, SHA-384, a SHA-512 je ve srovnání s předešlými bezpečnější. Funkce SHA2() vytváří obraz délky 224, 256, 384 a 512 bitů, což ve srovnání se 160bitovým obrazem, který vrací funkce SHA1(), umožňuje zvýšit bezpečnost (mysql.com, 2012). To znamená, že současnými prostředky je nemožné v rozumně krátké době nalézt řetězec, který odpovídá obrazu, nebo nalézt odlišný řetězec s totožným otiskem (tzv. kolize) (DuBois, 2009).

MySQL přesto stále používá pro uložení hesla uživatele v tabulce *mysql.user* nedostačující funkci PASSWORD().

4.2. Zabezpečení DBMS

4.2.1. Protokolování

Protokolování úzce souvisí se zálohováním. Při větším objemu dat je úplné zálohování časově náročný proces a nelze ho provádět příliš často. Záznamy protokolů průběžně vytvářejí záznam všech změn provedených v databázi. Pro zabezpečení dat může být tento záznam následně využit společně s poslední úplnou zálohou při obnově poškozených nebo ztracených dat. Analýzou protokolů lze také určit kdo, kdy a jakým způsobem s databází pracoval a z toho vyvodit závěry. Příkladem je neobvykle vysoký počet neplatných pokusů přihlášení k serveru MySQL a z toho plynoucí bezpečnostní rizika. V případě pokusu o prolomení hesla následuje omezení dostupnosti pro daného uživatele. Nastavení jednotlivých druhů protokolů jsou uložena v konfiguračním souboru *my.cnf*. Samotné soubory protokolů jsou uloženy ve stejných adresářích společně s příslušnými databázemi. Pro zvýšení bezpečnosti a rychlosti serveru je vhodné uložit protokoly na jiný pevný disk (Kofler, 2007).

Protokolování aktualizací

Pro zaznamenávání všech příkazů provádějících změny v datech je použit binární protokol nastavený direktivou `log-bin = jméno_souboru`. Binární soubor je omezen na velikost `max_binlog_size = n`, která je defaultně nastavena na 1 GB. Binární soubory jsou ukládány ve tvaru *jméno_souboru.n*, kde *n* je celé číslo, respektive pořadí souborů rozdělených podle hodnoty max_binlog_size. Protokolování lze omezit jen na konkrétní databázi (`binlog_do-db = jméno_databáze`), nebo lze konkrétní databázi vyjmout z protokolování (`binlog_ignore-db = jméno_databáze`) (Kofler, 2007).

Protokolování činnosti serveru

V obecném logu jsou zaznamenány události připojení, odpojení a všechny dotazy vykonané i nevykonané. Zapnutí obecného logu se provádí uložením záznamu `log = jméno_souboru`. Tento log zaznamenává veškeré činnosti bez ohledu na to, zda jsou vykonány nebo ne. Neslouží tedy k obnově dat (Kofler, 2007).

Protokolování pomalých dotazů

Protokolování pomalých dotazů je záznam vykonaných pomalých dotazů, které trvaly déle než je stanovený časový limit. Zapnutím logu pomalých dotazů direktivou `log-slow-queries = jméno_souboru` můžeme dále upřesnit vlastnosti záznamu logu. Direktiva `long_query_time = t`, kde *t* je čas ve vteřinách, určuje čas, po jejímž překročení je dotaz zaznamenán. Záznam pomalých dotazů umožňuje zaznamenávat i dotazy, ve kterých nejsou použity indexy nastavením direktivy `log_queries_not_using_indexes`. Příkladem tohoto dotazu bez použití indexu je dotaz `SELECT * FROM tabulka;`, výběr všeho. Tyto dotazy jsou zpravidla rychlé, efektivní a časté. Jejich záznam je důležitý při vyhodnocování optimalizace výkonu serveru (Kofler, 2007; Schwartz, 2009).

Protokolování chyb

Zaznamenávání protokolu chyb nelze vypnout, lze pouze nastavit název souboru direktivou `log-error = jméno_souboru`. Není-li toto nastavení uloženo, je soubor pojmenován podle názvu hosta (*název_hosta.err*). Kromě chybových hlášení jsou zaznamenávána spuštění a ukončení serveru (Kofler, 2007).

Manipulace se soubory protokolů

Správa jednotlivých protokolů je nejednotná. Příkaz `mysqladmin flush-logs` zavře soubory protokolů aktualizací a vytvoří nové s následujícím pořadovým číslem. Soubor protokolu chyb se přejmenuje z *název_hosta.err* na *název_hosta.old*. Protokoly činnosti serveru a pomalých dotazů jsou tvořeny jedním souborem. Při zálohování těchto protokolů za běhu MySQL serveru je nutné soubory přejmenovat a poté spustit příkaz `mysqladmin flush-logs`. S nově přejmenovanými soubory protokolů lze dále pracovat, zálohovat (Kofler, 2007). Podrobný návrh automatického přejmenování je v kapitole 4.2.3 Návrh rotace protokolů.

4.2.2. Zálohování

Zálohování neoddělitelně souvisí se zotavením. Samotný proces zotavení je neméně důležitý. Jedná se o kritický proces, na který jsou kladeny vysoké požadavky spolehlivosti a rychlosti. Schwartz (2009) uvádí, že samotný proces zálohování je mnohem snadnější než zotavení. Zálohování je i z hlediska bezpečnosti a šifrování rutinní záležitost, není prováděno pod extrémním tlakem (jako krizová situace zotavení).

Rozlišujeme zálohy online a offline. Je-li možné MySQL server vypnout pro pořízení offline zálohy, získá se přesná kopie dat. Při použití úložného enginu InnoDB s objemnými datovými celky dochází k nárůstu času potřebného k uložení dat z paměti na disk (tzv. „spláchnutí“, příkaz FLUSH). I následné spuštění serveru v plném výkonu je u velkého množství tabulek časově náročné (tzv. proces zahřátí cache). V případech, kde není možné server zcela odstavit, lze uplatnit postup, který cituje Schvartz (2009). Velice často používaný postup pro tabulky MyISAM začínající příkazem FLUSH TABLES WITH READ LOCK uloží a uzamkne všechny tabulky společně s cache dotazů. Tento postup je méně náročný než pořízení offline zálohy, ale z důvodu času ne vždy uplatnitelný. Doba provedení, po kterou nelze ukládat do uzamčených tabulek, je ovlivněna čekáním na dokončení nejdéle trvajícího příkazu.

Online záloha vznikne využitím logování změn dat a změn datových struktur databáze. V uvedené online záloze je možné se vrátit k jakémukoli předešlému okamžiku, tedy stavu databáze, ve kterém se v daný okamžik nacházela – PITR (Point In Time Recovery) (Valenta, 2009).

Dále rozlišujeme logické zálohy a zálohování souborů. Logické zálohy jsou tvořeny příkazy a daty SQL, nebo čistě jako prostý text s oddělovači. Zálohování souborů je prováděno kopírováním souborů uložených na disku.

Postup vytváření záloh

První fáze je tvořena volbou vhodného typu logických záloh. Nároky kladené na databázový server jsou rozhodující při volbě mezi Dumpy SQL a Dumpy, ve formátu s oddělovači. Jedná se o rychlost vytvoření zálohy a zejména rychlost obnovení ze zálohy.

Dalším krokem je zvolit postup, plán manipulace se samotnými zálohami, tedy předejít nechtěnému přepsání staré zálohy novou. Podrobný postup této části je rozpracován v kapitole 4.2.3. Návrh rotace protokolů. A také plán zabezpečení záloh proti zcizení a neoprávněnému přístupu (DuBois, 2009). Zvláště protokoly obsahují citlivé informace: samotná data a přístupová hesla uživatelů apod. Důraz na tento bod by měl být kladen obzvláště při zálohování na jiná záložní zařízení.

Úložné enginy - ACID

Úspěšnost zálohování a možnost obnovy závisí na použitých úložných enginech. V prostředí vyžadujícím transakce shrnují požadavky na úložné enginy kritéria ACID[31]. Jedná se o soubor vlastností umožňujících provádění transakcí. Transakce je nedělitelná (atomicita). Je provedena jako celek, nebo se vůbec neprovede, a během provádění transakce nedojde k narušení integritních omezení (konzistence). Při provádění transakce je pro ostatní operace transakce skryta (izolovanost). Úspěšně ukončené transakce jsou uloženy v databázi (trvalost).

4.2.3. Návrh rotace protokolů

MySQL zapisuje log soubory stále se stejným jménem. Snahou zálohování je v pravidelných intervalech tyto soubory přejmenovat a zálohovat. Četnost vytváření, dělení těchto logů se odvíjí od vytížení serveru a stupně zabezpečení. U velmi vytížených serverů narůstá rychle velikost protokolů. Pro vyšší zabezpečení je nutné provádět zálohu protokolů v co nejkratších intervalech, čímž vzniká téměř spojitá řada bodů obnovy – PITR.

[31] ACID – Atomicity Consistency Isolation Durability - atomicita, konzistence, izolovanost a trvanlivost.

Operační systém UNIX umožňuje přejmenování souboru logu, i když je serverem otevřen. Poté se provede příkazem FLUSH zapsání dat z operační paměti do otevřeného logu a jeho zavření. Server následně otevře nový soubor a pokračuje v ukládání do nového souboru. Uvedený postup je aplikován na rotaci do sedmi souborů, kdy vždy nejmladší log má nejnižší číslo. Vzorový skript je uveden v příloze č. 2: Shell script pro rotaci souborů logu (DuBois, 2009). Parametrem, s kterým je tento skript spouštěn, je jméno (případně cesta) souboru logu. Při každém spuštění tohoto skriptu dojde k přejmenování, tedy rotaci názvů jednotlivých logů. Pro rotaci *glogu* umístěného v */usr/mysql/data/* je skript spuštěn:

```
% rotaceFixnichLogu.sh /usr/mysql/data/glog
```

Script je vytvořen pro spuštění z účtu *mysql*, oprávnění je tedy nastaveno:

```
% chmod go-rwx rotaceFixnichLogu.sh
```

V MySQL je vytvořen účet, který má přiděleno oprávnění pouze provádět příkaz FLUSH. Pod tímto uživatelem bude spouštěn příkaz FLUSH volaný z uvedeného scriptu.

```
CREATE USER 'flush'@'localhost' IDENTIFIED BY 'flushpass';

GRANT RELOAD ON *.* TO 'flush'@'localhost';
```

Na systémech Unix lze skript spouštět periodicky službou *cron*. V rámci dalšího zvyšování zabezpečení lze postupovat až k úplnému oddělení souborů protokolů, například na jiný disk, nebo zcela na jiný server apod.

4.2.4. Omezení serveru limity připojení

Bezpečnost databázového serveru lze ovlivnit vhodnou konfigurací maximálního počtu chybných, tj. neuskutečněných připojení. Tedy takových připojení, u kterých nedojde k úspěšnému autorizování uživatele, nebo pokud nedojde k úspěšnému dokončení prováděné relace. Při dosažení nastaveného počtu takovýchto pokusů dochází k blokování připojení z daného hostitele.

Nastavení proměnné *max_connection_errors* je uloženo v konfiguračním souboru *my.cnf*. Hodnota této proměnné je nenulová. Nelze tedy kontrolu zcela vypnout. Pro eliminaci funkce tohoto omezení lze nastavit velmi vysoké číslo. Překročení, tedy zablokování uživatele je uloženo do logu spolu s informací o příkazu pro odblokování (Kofler, 2007).

```
Host 'host.spatny.com' blocked because of many connection errors.
Unblock with 'mysqladmin flush-hosts'
```

Odblokování příkazem mysqladmin *flush-hosts* vyprázdní vyrovnávací paměť pro všechna jména hostitelů, nelze tedy odblokovat pouze jednoho hostitele (Schwartz, 2009).

4.2.5. Omezení síťových služeb serveru

Je-li MySQL server provozován v prostředí nevyžadujícím přístup přes síť, jedná se tedy o stejného hostitele, je z důvodu bezpečnosti nutné vypnout síťový přístup. Možnosti vzdálené správy MySQL serveru jsou omezeny pouze na vzdálený SSH přístup, nebo skrze webové rozhraní, například aplikace *phpMyAdmin, SQL Buddy*. Právě i instalace a použití aplikace *phpMyAdmin* skýtá bezpečnostní riziko. Nastavení zákazu síťové podpory je provedeno v konfiguračním souboru *my.cnf*, přidáním příkazu `skip_networking`. Tato konfigurace omezuje použití například Java aplikací připojujících se skrze TCP/IP – Connector/J. Je tedy nutné omezit TCP připojení pouze na připojení z lokálního serveru. V takovém případě je namísto *skip_networking* vložen kód `bind_address = 127.0.0.1`, povolující pouze připojení z „localhost“ (Schwartz, 2009).

4.3. Zabezpečení proti nežádoucím přístupům

4.3.1. Bezpečnostní rizika ve zdrojovém kódu aplikace - SQL Injection

SQL Injection a Cross Site Scripting[32] (XSS) tvořily za 3. a 4. čtvrtletí 2010 největší podíl útoků na komerční aplikace (Cenzic, 2010). Podle přílohy č. 1: Zranitelnost webu podle druhu útoků zaujímá XSS 36% podíl a SQL Injection 18% podíl z celkových útoků za uváděné období. SQL Injection je technika pokoušející se pozměnit původní SQL příkaz za účelem získání nebo pozměnění dat databáze. Vstup škodlivého kódu může být obdobný XSS skrze neošetřený vstup a výstup ve zdrojovém kódu. Obsahuje SQL příkaz, který je vložen v části skriptu provádějící dotaz nad databází. Příkladem je SQL příkaz, který ukončí předešlý příkaz v rámci kterého je vkládán, a připojí vlastní SQL příkaz, který je škodlivý. Ten poté

[32] Cross Site Scripting (XSS) – napadení webové stránky skrze chybu ve skriptu, neošetření vstupu a výstupu. Příkladem obrany je ošetření převodu „nebezpečných“ znaků na dané HTML entity.

může poškodit strukturu dat, odstranit některá data, či vyhodnotit podmínku celého dotazu kladně, viz příklad přihlášení uživatele - výňatek SQL dotazu v PHP:

```
SELECT * FROM uzivatele

WHERE login='"$login"' AND passwd='"$passwd"'
```

Pokud by proměnné *$login* a *$passwd* nebyly ošetřeny a byla by zadána proměnná:

```
$passwd =  "' OR  '"1"' = '"1
```

Byl by proveden následující SQL příkaz:

```
SELECT * FROM uzivatele

WHERE login='"$login"' AND passwd='""' OR  '"1"' = '"1"'
```

Výsledkem tohoto příkazu je vždy TRUE. Tedy přístup - přihlášení uživatele povoleno. Zabránit uživateli pozměnit SQL dotaz lze na úrovni databázové vrstvy MySQLi[33] oddělením SQL dotazu a uživatelských dat. Uživatelská data se předají jako parametry (Vrána, 2012).

```
<?php
    $stmt = $mysqli->prepare("SELECT * FROM tabulka WHERE nazev = ?
    OR id = ?");

    $stmt->bind_param("si", $_GET["nazev"], $_GET["id"]);

    $stmt->execute();
?>
```

(Vrána, 2005)

Existují další dvě možnosti obrany proti SQL Injection. Všechna vstupní data lze ošetřit PHP funkcí *mysql_real_escape_string()*, případně vlastní obdobnou funkcí, která ošetří všechny kritické znaky. Druhou možností je využití direktivy *magic_quotes_gpc* a uzavření všech hodnot do apostrofů. Vrána (2005) uvádí tento příklad:

[33] MySQLi – rozšíření umožňující využívat funkce MySQL 4.1 a vyšší.

```
<?php
    // ošetření vstupních dat
    mysql_query("SELECT * FROM tabulka WHERE nazev = '" .
    mysql_real_escape_string($_GET["nazev"]) . "' OR id = " .
    intval($_GET["id"]));
    // spolehnutí se na magic_quotes_gpc
    mysql_query("SELECT * FROM tabulka WHERE nazev = '$_GET[nazev]'
    OR id = '$_GET[id]'");
?>
```
(Vrána, 2005)

4.4. Adresářová struktura MySQL

Znalost adresářové struktury je využitelná pro vhodnou bezpečnostní konfiguraci a také pro využití všech možností zálohování.

4.4.1. Přemístění datového adresáře

Výchozí nastavení není zejména při větším zatížení serveru výhodné. MySQL umožňuje změnit místo uložení jednotlivých datových souborů databáze. Důvodem proč změnit umístění souborů je například rozložení zátěže, nedostatek místa na disku a zvýšení bezpečnosti. Uložení databáze a log souborů na jiné fyzické disky vede ke snížení zatížení jednotlivých disků a k zajištění vyšší bezpečnosti při poruše disku. Obdobně lze rozdělit na více disků i jednotlivé oddíly tabulek.

Metody přemístění datového adresáře

Pro změnu umístění datového adresáře, nebo jeho částí, existují dva způsoby. Prvním obecným postupem je konfigurace souboru *my.cnf* přidáním:

```
[mysqld]
datadir = název_adresáře
```

Druhým způsobem, který podporuje systémem UNIX, je možnost změny umístění souborů a adresářů symbolickým linkem (tzv. symlink). Použití uvedených metod je v tabulce č. 1: Metody změny umístění adresářů a jejich částí.

Přesouvaný subjekt	Použitá metoda
Celý datový adresář	Konfigurační soubor, symlink
Samostatný adresář databáze	Symlink
Samostatná tabulka databáze	Symlink
Soubory tabulkového prostoru InnoDB	Konfigurační soubor
PID soubory serveru	Konfigurační soubor
Soubory Logů	Konfigurační soubor

Tabulka č. 1: Metody změny umístění adresářů a jejich částí (DuBois, 2009, s. 603)

4.5. Securich

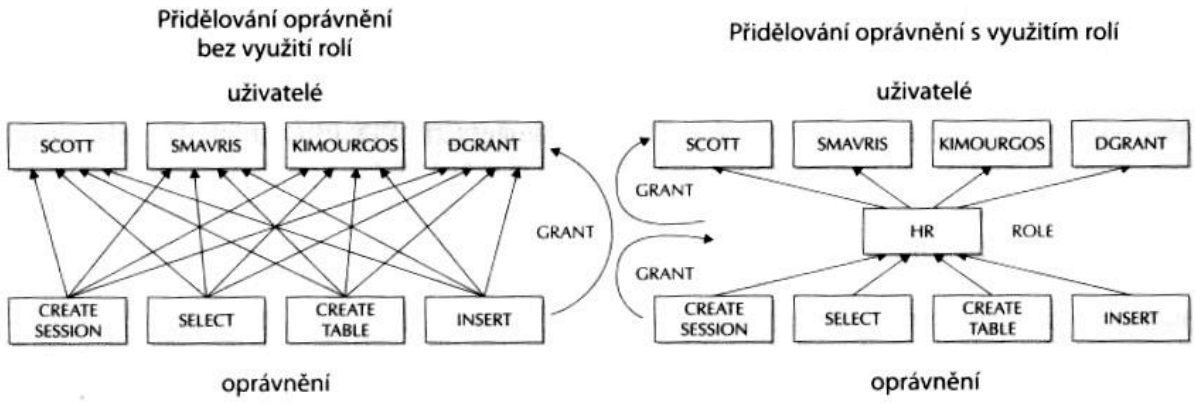
Plugin Securich je kompatibilní s verzí MySQL 5.1.12 a novějšími. Jedná se o Open source projekt. Securich zefektivňuje práci s hesly, uživateli a správou databáze. Pro práci s hesly přidává možnost nastavení délky hesla, historii hesla a minimální stáří hesla. Pro práci s uživateli umožňuje přidělování oprávnění skrze pseudorole a klonování uživatelů. Umožňuje přidávání a odebírání oprávnění nad tabulkami s využitím regulárních výrazů (Cassar, 2011).

V současné době je projekt už rok ve verzi 0.6.2 (květen 2011) a informace o směřování dalšího vývoje jsou nedostupné. Vyjádření autora Darrena Cassara k budoucímu vývoji tohoto projektu se nepodařilo zjistit.

4.6. Správa oprávnění v DBMS a absence rolí v MySQL

Role umožňují zjednodušit a značně zefektivnit úlohu správce databáze. Přidělování systémových a objektových oprávnění každému uživateli zvlášť je při větším počtu uživatelů časově náročné a i rizikové kvůli vzniku možné chyby. V komerčních databázích (Oracle, Microsoft SQL Server) jsou využívány pro správu oprávnění tzv. *role*, neboli pojmenované množiny oprávnění pro určitou skupinu uživatelů. Grafické srovnání DBMS nevyužívajících, respektive využívajících přidělování oprávnění s využitím rolí je znázorněno na obrázku č. 4: Použití rolí pro správu oprávnění. Na uvedeném obrázku je graficky znázorněno množství příkazů *grant, revoke* mezi jednotlivými uživateli a přidělenými právy. Ve srovnání DBMS

bez integrace rolí (např. MySQL) a systémů využívajících role, je správa oprávnění s rolemi značně jednodušší a přehlednější. Pozdější změny oprávnění pro celou skupinu uživatelů vyžadují pouze úpravu množiny oprávnění příslušné role. Systém ORACLE umožňuje uživatelům aktivovat role automaticky při přihlášení uživatele, nebo využít ochrany rolí heslem. Ochrana rolí heslem umožňuje rozšíření autentizačního schématu přihlašování uživatelů o novou úroveň zabezpečení (Loney, 2010).



Obrázek č. 4: Použití rolí pro správu oprávnění (Loney, 2010, s. 336)

5. Zhodnocení výsledků

Problematika zabezpečení MySQL je komplexní problém přesahující samotný DBMS. Práce se zabývá přehledem oblastí, které ovlivňují bezpečnost DBMS. V oblastech přístupu a zabezpečení MySQL jsou zdůrazněny bezpečnostní rizika, která vyplývají z nevhodné konfigurace, nebo nevyužití všech možností, které MySQL nabízí. Oblast zabezpečení přístupů přesahuje MySQL a zasahuje do konfigurace serveru Apache, jež předpokládá využívání bezpečné konfigurace přístupu k souborům v adresářové struktuře webového serveru a zejména souborů obsahujících přístupové údaje k databázi. Na straně databáze jsou klíčové zásady udělování oprávnění, zvláště pak vyloučení anonymního uživatele v samotné konfiguraci MySQL. A také správné použití příkazu pro odebrání oprávnění a odstranění uživatelů. Příkaz REVOKE pouze odstraňuje oprávnění, uživatel se stále může k databázi připojit – účet je úplně odstraněn příkazem DROP USER. Dalším častým rizikem jsou oprávnění udělená všem uživatelům. Zejména udělená automaticky po instalaci k databázi *test*, respektive k databázi s prefixem *test_*. Toto oprávnění není viditelné ve výsledcích příkazu SHOW GRANTS. Pro vzdálený přístup k serveru autor doporučuje striktně využít zabezpečených přístupů SSL a SSH. Významným bezpečnostním nedostatkem MySQL je použití v současnosti již nedostačující kryptografické funkce PASSWORD() pro uložení hesla uživatele.

V oblasti samotného zabezpečení DBMS je kladen důraz na vhodné zvolení způsobu zálohování databáze, závisející na podmínkách provozu databáze a očekávaného způsobu zotavení. Pro dosažení konzistentní zálohy obsahující jednotlivé body obnovy (PITR) je nutné vhodně využít kombinaci protokolování změn dat a datových struktur společně s vhodnými úložnými enginy splňujícími kritéria ACID. Autor zde uvádí postup řešení rotace souborů protokolů skriptem shellu. Stabilita databázového serveru také závisí na nastavení limitů pokusů o připojení k serveru. Základním předpokladem pro zabezpečení serveru MySQL je konfigurace síťových služeb podle provozu serveru. Doporučením je lokální přístup, se striktně zakázaným síťovým přístupem.

Obranou proti nežádoucím přístupům je odstranění časté chyby ve zdrojovém kódu aplikace umožňující SQL Injection.

V souvislosti se zálohováním autor shrnuje metody umožňující přemístění datového adresáře v Unixovém operačním systému. Využití těchto metod společně s dalšími metodami

ochrany fyzických dat (RAID pole) nejen zvyšuje bezpečnost, ale také výkonnost celého serveru.

Úložné procedury umožňují rozšířit a zjemnit správu oprávnění pro konkrétní podkladové tabulky. Zároveň ale přinášejí riziko spojené s udělením oprávnění k těmto úložným procedurám, úložné procedury mají totiž oprávnění tvůrce úložné procedury.

Používání rolí při správě přístupů v komerčních DBMS umožňuje tuto práci zefektivnit a snížit riziko vzniku chyb. Absence tohoto přístupu v MySQL je nedostatkem, který se snaží odstranit projekt Securich. Doplňuje správu oprávnění, přidání rolí a zlepšuje práci s hesly. Budoucí směřování tohoto projektu je ale nejisté a z tohoto důvodu by nebylo vhodné „ostré“ nasazení tohoto doplňku.

6. Závěr

Způsoby správy přístupů pro MySQL obsahují riziko spojené s větším počtem uživatelů. Při správě uživatelů dochází ke snížení přehlednosti v důsledku absence rolí ve srovnání s konkurenčními DBMS. Autor dále upozornil na vhodné použití příkazů pro správu uživatelů při odstraňování oprávnění příkazy REVOKE a DROP USER. Po instalaci MySQL autor také doporučuje vždy odstranit přístupová práva k databázi *test*, respektive k databázi s prefixem *test_*. Při pozdější kontrole může dojít k opomenutí, neboť záznam o těchto oprávněních není ve výsledcích příkazu SHOW GRANTS. Možnosti zabezpečení přístupů je vhodné rozšířit o vzdálený přístup prostřednictvím certifikátů SSL a připojení SSH. Nedostatek MySQL autor shledává v samotném zabezpečení hesla uživatelů v kontextu s dnes používanými kryptografickými algoritmy.

V oblasti zabezpečení v druhé části této práce je pozornost kladena na možnosti zálohování a následný proces zotavení. Jsou uvedeny možnosti dosažení konzistentní zálohy obsahující jednotlivé body obnovy s vhodným návrhem manipulace se soubory protokolů. Pro udržení stability serveru autor doporučuje vhodné nastavení limitů pokusů o připojení. Také zmiňuje striktně zakázat síťový přístup při lokálním použití serveru. Bezpečnostním rizikem ze strany webových aplikací je stále častý útok na proniknutí do databáze - SQL Injection. Autor v této práci uvádí příklad obrany proti tomuto útoku na úrovni PHP skriptu.

Problematika zabezpečení databáze MySQL je komplexní problém přesahující samotný DBMS, zasahující do konfigurace webového serveru i firewallu.

7. Seznam použitých zdrojů

BARRETT, D. J., Silverman, R. E. *SSH Kompletní průvodce.* Brno: Computer Press, a.s., 2003. ISBN 80-7226-852-X.

BŘÍZA, P. *Znakové sady v praxi - UTF-8.* [online]. c2012, [cit. 02-01-2012]. Dostupné z WWW: <http://interval.cz/clanky/znakove-sady-v-praxi-utf-8>.

CASSAR, D. *Securich*. [online]. c2011, [cit. 02-10-2012]. Dostupné z WWW: <http://code.google.com/p/securich/wiki/Documentation>.

CODD, F. E. *A Relational Model of Data for Large Shared Data Banks.* [online]. c1970, [cit. 11-14-2011]. Dostupné z WWW: <http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>.

CENZIC. *Web Application Security Trends Report Q3-Q4, 2010.* [online]. c 2012, [cit. 02-10-2012]. Dostupné z WWW: <http://www.cenzic.com/downloads/Cenzic_AppSecTrends_Q3-Q4-2010.pdf>.

CHAMBERLIN, D. D., Boyce, R. F. *SEQUEL: A STRUCIURED ENGLISH QUERY LANGUAGE.* [online]. c1974, [cit 10-05-2011]. Dostupné z WWW: <http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf>.

CONOLLY, T. *Mistrovství - databáze.* Brno: Compuer press, a. s., 2009. ISBN 978-80-251-2328-7.

COOPER, M. *Internet X.509 Public Key Infrastructure*. [online]. c2005, [cit. 02-10-2012]. Dostupné z WWW: <http://tools.ietf.org/html/rfc4158>.

DUBOIS, P. *MySQL Developer's Library.* Boston: Addison-Wesley, 2009. ISBN 978-0-672-32938-8.

Drupal 7 - Instalace. [online]. c2012, [cit. 02-20-2012]. Dostupné z WWW: <http://navody.c4.cz/drupal-7-instalace>.

GILMORE, J. W., *Velká kniha PHP a MySQL 5: kompendium znalostí pro začátečníky i profesionály.* Brno: Zoner Press, 2007. ISBN 978-80-86815-53-4.

HAVRLANT, L. *matweb.cz*. [online]. c2011, [cit. 10-20-2011]. Dostupné z WWW: <http://www.matweb.cz/relace>.

HÁLA, T. *Active24.cz.* [online]. c2012, [cit. 02-09-2012]. Osobní komunikace.

Instalace webu jedním kliknutím. [online]. c2012, [cit. 02-20-2012]. Dostupné z WWW: <http://www.active24.cz/o-spolecnosti/akce/instalace-webu-jednim-kliknutim>.

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

LONEY, K. *Oracle Database Kompletní průvodce.* Brno: Computer Press, a.s., 2010. ISBN 978-80-251-2489-5.

MACH, J. *Cesky-hosting.cz.* [online]. c2012, [cit. 03-13-2012]. Osobní komunikace.

Mysql.com. *11.13. Encryption and Compression Functions*. [online]. c2012, [cit. 02-20-2012]. Dostupné z WWW: <http://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html>.

POKORNÝ, J. *Dotazovací jazyky.* Veletiny: SCIENCE, 1994. ISBN 80-901475-2-6.

POKORNÝ, J. *Databázové systémy 2.* Praha: ČVUT, 2007. ISBN 978-80-01-03797-3.

RIORDAN, R. M. *Vytváříme relační databázové aplikace.* Praha: Computer Press, 2000. ISBN 978-80-7226-360-8.

ROELOFS, G., Adler, M. *zlib.net.* [online]. c2012, [cit. 03-15-2012]. Dostupné z WWW: <http://zlib.net>.

ROSEBROCK, E., Filsone, E. *Linux, Apache, MySQL a PHP.* Praha: GRADA, 2005. ISBN 80-247-1260-1.

SCHNEIDERX, Robert. *MySQL Oficiální průvodce tvourbou, správou a laděním databází.* Praha: Grada Publishing, a.s., 2009. ISBN 80-247-1516-3.

SCHWARTZ, B., ZAITSEV, P., TKACHENKO, V., ZAWODNY, J., LENTZ, A., BALLING, D.. *MySQL profesionálně - optimalizace pro vysoký výkon.* Brno: Zoner software, a.s., 2009. ISBN 978-80-7413-035-9.

System requirements. [online]. c2012, [cit. 02-12-2012]. Dostupné z WWW: <http://drupal.org/requirements>.

VALENTA, M. *Zabezpečení db.* [online]. c2012, [cit. 03-22-2012]. Osobní komunikace.

VALENTA, M. *DBA - MySQL.* [online]. c2012, [cit. 02-05-2012]. Dostupné z WWW: <http://service.felk.cvut.cz/courses/Y36DBA/slides-2009/dba_04_mysql.pdf>.

VOSTROVSKÝ, V. *Vytváření databází v Oracle.* Praha: Reprografické studio PEF ČZU v Praze, 2004. ISBN 978-80-213-1191-6.

VRÁNA, J. *Obrana proti SQL Injection.* [online]. c2005, [cit. 02-12-2012]. Dostupné z WWW: <http://php.vrana.cz/obrana-proti-sql-injection.php>.

Vysoká škola báňská - Technická univerzita Ostrava, *MSSQLServer: 3. Architektura databází.* [online]. c2012, [cit. 02-07-2012]. Dostupné z WWW: <http://www.fs.vsb.cz/books/MSSQLServer/MSSQL_soubory/SQL_index3.htm>.

WALTERS, R. E. *Mistrovství v Microsoft SQL Server 2008.* Brno: Computer Press, a.s., 2009. ISBN 978-80-251-2329-4.

ZENDULKA, J. *Databázové systémy: 10. Architektura klient / server a třívrstvá architektura.* [online]. c2012, [cit. 02-05-2012]. Dostupné z WWW: <http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/10_2.pdf>.

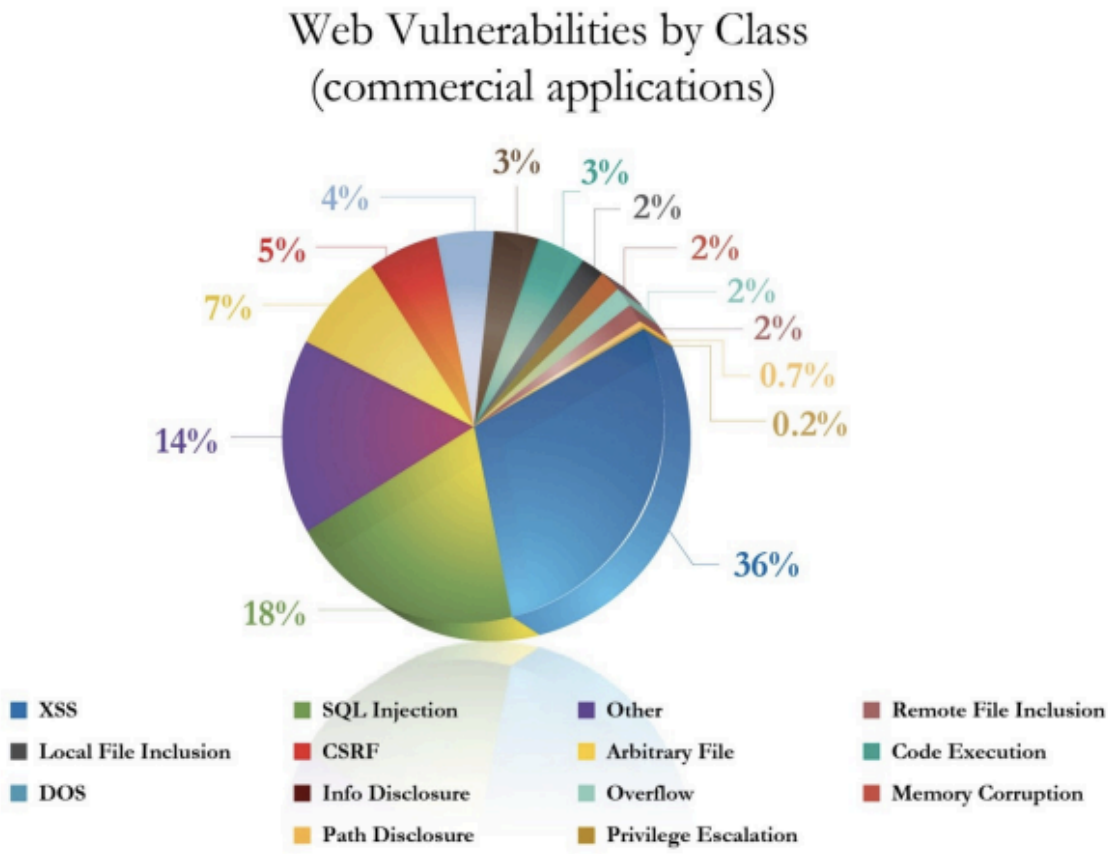
8. Přílohy

8.1. Seznam obrázků

8.2. Seznam tabulek

8.3. Seznam příloh

Příloha č. 1: Zranitelnost webu podle druhu útoků (Cenzic, 2010)



Příloha č. 2: Shell script pro rotaci souborů logu (DuBois, 2009)

```
1.  #!/bin/sh
2.  # rotaceFixnichLogu.sh - rotace souboru logů se stálým jménem
3.  # Argument 1: log jmenoSouboru
4.  if [ $# -ne 1 ]; then
5.  echo "Pouzit: $0 jmenologu" 1>&2 exit 1
6.  fi
7.  logfile=$1
8.  mv $logfile.6 $logfile.7
9.  mv $logfile.5 $logfile.6
10. mv $logfile.4 $logfile.5
11. mv $logfile.3 $logfile.4
12. mv $logfile.2 $logfile.3
13. mv $logfile.1 $logfile.2
14. mv $logfile $logfile.1
15. mysqladmin -u flush -pflushpass flush-logs
```

Příloha č. 3: Soubor *my.cnf*

```
#
# The MySQL database server configuration file.
#
# You can copy this to one of:
# - "/etc/mysql/my.cnf" to set global options,
# - "~/.my.cnf" to set user-specific options.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
    # --print-defaults to see which it would actually understand and
use.
    #
    # For explanations see
    # http://dev.mysql.com/doc/mysql/en/server-system-variables.html
    # This will be passed to all mysql clients
    # It has been reported that passwords should be enclosed with
ticks/quotes
    # escpecially if they contain "#" chars...
    # Remember to edit /etc/mysql/debian.cnf when changing the
socket location.


    [client]
    port       = 3306
    socket          = /var/run/mysqld/mysqld.sock
    # Here is entries for some specific programs
    # The following values assume you have at least 32M ram
    # This was formally known as [safe_mysqld]. Both versions are
currently parsed.
    [mysqld_safe]
    socket          = /var/run/mysqld/mysqld.sock
    nice            = 0


    [mysqld]
    #
    # * Basic Settings
    #
    user       = mysql
    pid-file   = /var/run/mysqld/mysqld.pid
    socket     = /var/run/mysqld/mysqld.sock
    port       = 3306
    basedir    = /usr
    datadir    = /var/lib/mysql
    tmpdir     = /tmp
    language   = /usr/share/mysql/english
    skip-external-locking
    #
    # Instead of skip-networking the default is now to listen only
on
    # localhost which is more compatible and is not less secure.
    bind-address         = 127.0.0.1
    #
    # * Fine Tuning
    #
    key_buffer           = 16M
    max_allowed_packet   = 16M
    thread_stack         = 192K
    thread_cache_size    = 8
```

```
51.     # This replaces the startup script and checks MyISAM tables if
   needed
52.     # the first time they are touched
53.     myisam-recover         = BACKUP
54.     #max_connections        = 100
55.     #table_cache            = 64
56.     #thread_concurrency     = 10
57.     #
58.     # * Query Cache Configuration
59.     #
60.     query_cache_limit      = 1M
61.     query_cache_size       = 16M
62.     #
63.     # * Logging and Replication
64.     #
65.     # Both location gets rotated by the cronjob.
66.     # Be aware that this log type is a performance killer.
67.     # As of 5.1 you can enable the log at runtime!
68.     #general_log_file        = /var/log/mysql/mysql.log
69.     #general_log             = 1
70.     #
71.     # Error logging goes to syslog due to
   /etc/mysql/conf.d/mysqld_safe_syslog.cnf.
72.     #
73.     # Here you can see queries with especially long duration
74.     #log_slow_queries       = /var/log/mysql/mysql-slow.log
75.     #long_query_time        = 2
76.     #log-queries-not-using-indexes
77.     #
78.     # The following can be used as easy to replay backup logs or for
   replication.
79.     # note: if you are setting up a replication slave, see
   README.Debian about
80.     #       other settings you may need to change.
81.     #server-id              = 1
82.     #log_bin                = /var/log/mysql/mysql-bin.log
83.     expire_logs_days       = 10
84.     max_binlog_size        = 100M
85.     #binlog_do_db           = include_database_name
86.     #binlog_ignore_db       = include_database_name
87.     #
88.     # * InnoDB
89.     #
90.     # InnoDB is enabled by default with a 10MB datafile in
   /var/lib/mysql/.
91.     # Read the manual for more InnoDB related options. There are
   many!
92.     #
93.     # * Security Features
94.     #
95.     # Read the manual, too, if you want chroot!
96.     # chroot = /var/lib/mysql/
97.     #
98.     # For generating SSL certificates I recommend the OpenSSL GUI
   "tinyca".
99.     #
100.    # ssl-ca=/etc/mysql/cacert.pem
101.    # ssl-cert=/etc/mysql/server-cert.pem
102.    # ssl-key=/etc/mysql/server-key.pem
```

```
103.    [mysqldump]
104.    quick
105.    quote-names
106.    max_allowed_packet   = 16M


107.    [mysql]
108.    #no-auto-rehash # faster start of mysql but no tab completition


109.    [isamchk]
110.    key_buffer      = 16M


111.    #
112.    # * IMPORTANT: Additional settings that can override those from
   this file!
113.    #   The files must end with '.cnf', otherwise they'll be
   ignored.
114.    #
115.    !includedir /etc/mysql/conf.d/
```