Timothy Boronczyk

sitepoint

MySQL

OKAMŽITĚ

computer press

OVLÁDNĚTE MySQL ZA VÍKEND

MySQL Okamžitě

Vyšlo také v tištěné verzi

Objednat můžete na
www.computerpress.cz
www.albatrosmedia.cz

Timothy Boronczyk
MySQL Okamžitě – e-kniha

ALBATROS MEDIA a.s.

Obsah

O autorovi

Timothy Baranczyk žije ve městě Syracuse ve státě New York a pracuje jako vývojář ve společnosti ShoreGroup, Inc. Webovým technologiím se věnuje již od roku 1998. Svoje vysokoškolské vzdělání završil získáním titulu v oboru programování. Kromě toho je také držitelem certifikátu Zend Certified Engineer. Svůj volný čas, kterého ale nemá příliš mnoho, tráví rád s přáteli. S oblibou také hovoří esperantem a miluje spaní s nohama vyčnívajícíma z postele. Poměrně snadno jej rozptýlí jakékoliv blýskavé předměty.

Úvod

Lze říci, že s daty se v současné době potkáte všude: počínaje ručně naškrábanými nákupními seznamy a konče rozsáhlými sadami dat (zvanými „big data“) v datových centrech nadnárodních korporací. Korporace shromažďují tolik dat, kolik jim umožňují jejich technologické možnosti, a následně je využívají k formulování nových obchodních a podnikových strategií. Vědci studují data proto, aby s jejich pomocí našli odpovědi na otázky, které mohou zachránit další lidské životy, zlepšit naše prostředí a vysvětlit naše místo v celém vesmíru. Dokonce i průměrný člověk shromažďuje během svého života poměrně značné množství dat, k nimž patří jak údaje v domácím účetnictví, popisující finanční chování každého z nás, tak i telefonní čísla v seznamu, uloženém v chytrém telefonu. Ukládání a celková organizace dat se natolik usnadnily, že mnohé z databázových konceptů a algoritmů, díky nimž se všechny tyto aktivity staly vůbec možnými, dnes již považujeme za samozřejmé.

Tato kniha je úvodem do základních konceptů práce s relačními systémy řízení báze dat (RDBMS – Relational Database Management System), především tedy oblíbeným a volně šiřitelným systémem MySQL. Podobně jako další knihy z řady *Okamžitě* se i takto kniha snaží o to, aby vám poskytla určitý náskok v porozumění zvolené technologii. Velice rychle – a doufejme, že i bez velkých bolestí hlavy – se seznámíte se základy, díky čemuž budete moci dále pokračovat ve svém dalším studiu.

Jsem velmi vděčný za to, že mi byla nabídnuta možnost napsat tuto knihu. Co ji od ostatních knih v této řadě odlišuje, je, že se věnuje technologii, která se využívá jak při vývoji webových aplikací, tak i mimo něj. Tím samozřejmě nechceme říci, že MySQL není oblíbená u vývojářů zabývajících se tvorbou webových aplikací – právě naopak! Databáze se však používají i v mnoha dalších oblastech, což jsem se snažil postihnout svým výběrem témat.

Co je databáze?

Ačkoliv máme obecnou tendenci spojovat význam slova **databáze** s digitálním světem počítačů, tento pojem ve skutečnosti označuje jakoukoliv uspořádanou kolekci dat. To tedy znamená, že databáze může být jak digitální (elektronická), tak také fyzická. Typickou fyzickou databází je například i skříň stojící ve vaší domácí pracovně a obsahující složky, plné faktur a dalších účetních dokladů. Anebo jí může být i sbírka kuchařek, uložených na polici, majících stránky s osahanými rohy a přetékajících dodatečnými recepty, které jste si vystřihli z nějakých časopisů a založili do jednotlivých knih.

V digitálním světě jsou databáze členěny podle způsobu, jímž jsou v nich ukládána a uspořádávána data. Mezi běžné typy digitálních databází patří:

- **Prosté databázové soubory** – v nich jsou data ukládána sekvenčně, velice často pak ve formátu prostého textu. Tyto soubory lze sice poměrně snadno vytvářet a přidávat do

nich další data, avšak jejich používání je provázeno i několika nevýhodami. Vyhledávání dat v prostých databázových souborech je poměrně pomalé, tyto soubory mohou obsahovat i redundantní data, a navíc je možné je vcelku snadno poškodit. Příkladem takového typu databáze může být textový soubor vytvořený při hře Solitaire a obsahující informace o výsledcích jednotlivých hráčů.

- **Hierarchické databáze** – v nich jsou data ukládána ve stromové struktuře (pomocí vztahů rodič – dítě). Jedná se o vysoce uspořádané databáze, nabízející velice efektivní vyhledávání. Jejich nevýhodou však je poměrně obtížné procházení v případě, že neznáte použité vztahy. V průběhu času se může stát obtížnou i údržba vztahů mezi daty. Příkladem hierarchické databáze může být registr systému Windows.
- **Dokumentově orientované databáze** – v nich jsou data ukládána ve volné podobě, přičemž jsou indexována pomocí klíčových či hashovacích hodnot. Tyto databáze na jedné straně nabízejí poměrně dobrou škálovatelnost v rozsáhlých sítích, na straně druhé je však jejich používání spojeno i s mnoha problémy typickými pro prosté databázové soubory. Tyto databáze totiž mohou obsahovat redundantní data, nejsou v nich udržovány žádné vztahy mezi daty a vyhledávání dat může být pomalé. Redis a CouchDB představují poměrně oblíbené „NoSQL" databázové systémy, sloužící k řízení těchto typů databází.
- **Relační databáze** – v nich jsou data ukládána do tabulek tvořených řádky a sloupci. Do podobných tabulek mohou být uspořádány například i vytištěné ceníky či jízdní řády autobusů. Relační databáze podporují indexování velkých objemů dat, umožňující jejich rychlé načítání, nicméně vztahy (vazby) mezi jednotlivými tabulkami mohou být velice složité.

Nad většinou moderních digitálních databází se nachází **databázový server**, což je aplikace, která je speciálně navržena k řízení databází a je odpovědná za umožnění a řízení přístupu k uloženým datům. V těchto systémech totiž platí, že nikdy nepracujete přímo s daty. Namísto toho na server odesíláte požadavky na přidání nějakých dat, jejich změnu, výmaz či načtení. Následně server za vás provede požadované operace a výsledky vám předá. Kniha, kterou právě čtete, je zaměřena na MySQL, což je databázový server určený k řízení relačních databází.

Standardním jazykem používaným pro komunikaci s relačními systémy řízení báze dat je zhruba od poloviny 80. let minulého století jazyk **SQL** (Structured Query Language – strukturovaný dotazovací jazyk). Jazyk SQL sestává z příkazů pro vkládání, načítání a řízení dat, příkazů pro vytváření a údržbu tabulek, a dokonce i z příkazů pro správu databází. Jednotlivé příkazy mohou být rozděleny do kategorií či „podjazyků" daných jejich určením: příkazy sloužící pro práci se samotnými daty vytvářejí DML (Data Manipulation Language – jazyk pro manipulaci s daty); příkazy sloužící ke správě tabulek a databází vytvářejí DDL (Data Definition Language – jazyk pro definici dat); a nakonec příkazy sloužící k řízení přístupu k databázi vytvářejí DCL (Data Control Language – jazyk pro řízení dat).

Je dobré, abyste o tomto členění příkazů jazyka SQL věděli, neboť by se mohlo stát předmětem diskuze na nějaké party databázových administrátorů, které se třeba zúčastníte; já však v této knize takto podrobné členění používat nebudu. Jinými slovy řečeno, všechny příkazy jazyků DML, DDL a DCL budu souhrnně označovat jako SQL.

Od Edgara Codda k MySQL: stručná historie

V původních databázích byla data uspořádávána do stromových struktur. Přístup k takovým datům pak vyžadoval to, aby programátor napsal kód přímo procházející tyto struktury. Lze říci, že tento způsob práce s daty byl poměrně křehký a jakékoliv operace související s přidáváním dalších dat, aktualizacemi dat stávajících či změnami uspořádání dat byly dosti riskantní. Edgar Codd zkritizoval tento přístup ve své práci, nazvané *A Relational Model of Data for Large Shared Data Banks*, kterou vydal roku 1970. Tvrdil, že daleko lepší by bylo data uspořádat do tabulek a pracovat s nimi nezávisle na informacích o vzájemných vazbách, řazení a indexování. V té době se jednalo o velice zajímavý a inovativní koncept, kterého se chopila společnost IBM. Její inženýři ve vývojových laboratořích v San Jose začali pracovat na projektu s označením System R, který měl ověřit platnost teorií Edgara Codda.

Výsledkem projektu System R byla nejen první implementace jazyka SQL, ale také jasné důkazy potvrzující platnost relačních konceptů prosazovaných Edgarem Coddem. Když se Larry Ellison doslechl o výzkumu vedoucímu k prototypu System R, byl těmito informacemi tak zaujat, že myšlenky Edgara Codda a jazyk SQL začlenil do svého vlastního databázového serveru nazvaného Oracle. Mimochodem, při uvádění produktu na trh Larry Ellison porazil společnost IBM, a v roce 1979 se tak Oracle stal prvním komerčně dostupným relačním systémem řízení báze dat.

Mezitím se však o práci Edgara Codda začali zajímat i profesoři počítačových věd pracující na univerzitě státu Kalifornie se sídlem v Berkeley. Univerzitě se podařilo získat dostatek finančních zdrojů z National Science Foundation a z výzkumných divizí vojenského letectva a pozemních vojsk armády USA. Díky tomu byla univerzita schopna vytvořit rotující tým studentů vedený Michaelem Stonebrakerem a pracující na projektu INGRES. Tento projekt podrobně prozkoumal mnohé z relačních myšlenek publikovaných Edgarem Coddem, avšak kromě nich implementoval i vlastní dotazovací jazyk, nazvaný QUEL. Současně s tím, jak jednotliví studenti dokončovali svoje studia a odcházeli pracovat do jiných softwarových společností, se na trhu začaly objevovat databázové systémy inspirované projektem INGRES. K těm nejzajímavějším patřil Sybase, jehož licenci si později zakoupila společnost Microsoft. Ta jej pak přejmenovala na Microsoft SQL Server. I samotný INGRES byl nakonec uveden na trh v komerční podobě a poměrně rychle zaujal na trhu vůdčí pozici.

Ta se však začala otřásat zhruba okolo roku 1985, kdy počítačová veřejnost začala upřednostňovat jazyk SQL před QUEL. A navíc roku 1987 byl jazyk SQL přijat jako standard, a to jak americkou organizací ANSI (American National Standards Institute), tak i mezinárodní

organizací ISO (International Organization for Standardization). Na konci této dekády pak trhu jasně vévodil databázový systém Oracle spolu s jazykem SQL.

V roce 1993 David Hughes pracoval na aplikaci pro monitorování sítě, která si ukládala data do databáze řízené systémem Postgres (což byl jeden z nástupců systému INGRES). Z důvodu přenositelnosti však chtěl zajistit přístup k těmto datům i prostřednictvím jazyka SQL, a proto napsal překladač z jazyka QUEL do jazyka SQL, který nazval miniSQL. Během dalšího vývoje této monitorovací aplikace však byl David Hughes čím dál více zklamán hardwarovými nároky databázového systému Postgres, a proto se rozhodl, že na základě miniSQL vyvine svůj vlastní hardwarově nenáročný systém řízení báze dat. Řešení miniSQL upřednostňovalo malé hardwarové nároky před striktním a úplným dodržováním standardů SQL, a proto v něm byly implementovány pouze ty nejdůležitější části tohoto standardu. David Hughes byl schopen distribuovat svůj systém za zlomek nákladů ostatních komerčně dostupných databázových systémů. Lze tedy říci, že miniSQL se stalo prvním nízkonákladovým relačním systém řízení báze dat založeným na standardu SQL. Vše bylo připraveno pro příchod MySQL.

V téže době Monty Widenius vyvíjel webovou aplikaci pro – v té době – bouřlivě se rozvíjející internet. Základem jeho aplikace byl databázový server UNIREG, který si sám vytvořil. Monty Widenius zjistil, že přístup k systému UNIREG při generování dynamických webových stránek vyžaduje příliš mnoho prostředků, a proto se začal poohlížet po alternativním řešení. Jeho pozornost zaujal systém miniSQL, který se stal velmi populárním především díky své cenové strategii, zejména pak mezi poskytovateli webhostingových služeb. Systém miniSQL však neimplementoval některé z funkcí, které Monty Widenius potřeboval. Z tohoto důvodu se Monty Widenius rozhodl přepsat svůj systém UNIREG tak, aby zajišťoval vyšší výkon. Současně však využil tuto příležitost i k tomu, aby reimplementoval API systému UNIREG tak, aby nabízelo kompatibilitu s API systému miniSQL. Díky tomu by pak mohl využít mnoho nástrojů třetích stran nabízených pro systém miniSQL. Monty Widenius přejmenoval takto upravený systém na MySQL a jeden z jeho přátel jej přesvědčil, aby jej zveřejnil.

Systém MySQL byl zpřístupněn pod licencí GNU General Public License a roku 1995 Monty Widenius spolu s Davidem Axmarkem a Allanem Larssonem založil společnost MySQL AB, jejímž cílem bylo vést další vývoj MySQL a nabízet alternativní licencování a podporu pro komerční zákazníky. A zatímco systém miniSQL byl cenově dostupný, systém MySQL byl pro většinu zákazníků prakticky zdarma.

Jelikož licenční podmínky MySQL umožňovaly začlenění tohoto systému do většiny distribucí Linuxu a jelikož API systému MySQL bylo kompatibilní s API miniSQL, přičemž ale nabízelo více možností, systém MySQL velice rychle zaujal tržní pozici miniSQL. Lze říci, že v současné době je systém MySQL druhým nejrozšířenějším relačním systémem řízení báze dat založeným na jazyku SQL (přičemž první místo zaujímá systém SQLite, a to především díky využití v chytrých telefonech a vestavěných systémech).

Alternativy a budoucnost MySQL

Roku 2008 společnost Sun Microsystens zakoupila společnost MySQL AB za cenu 1 miliardy USD a o 2 roky později Oracle Corporation získala společnost Sun Microsystems včetně veškerých jejích aktiv (tedy včetně MySQL) za cenu 7,4 miliardy USD. Tatáž společnost, která na konci 80. let minulého století porazila IBM a INGRES, se tak stala vlastníkem autorských práv k MySQL. Přitom společnost Oracle již měla svůj vlastní databázový systém, díky čemuž se veškeré obavy o budoucnost MySQL, které začala mít komunita ve chvíli, kdy se vlastníkem stala společnost Sun, pouze zesílily.

Avšak díky licenci GPL platí, že každý smí MySQL vzít a dále ji vylepšovat. Jedinou podmínkou je, že veškeré provedené změny musí mít odpovídající licenci. To znamená, že ostatní mohou databázový systém MySQL nejen rozšiřovat, ale dokonce mohou vyvíjet i svoje vlastní varianty (ve světě otevřeného softwaru nazývané forky) a veřejně je zpřístupňovat. A forků MySQL dnes existuje již celá řada:

- **Dorsal Source** – první fork MySQL, připravený společností Proven Scaling jako reakce na pomalý proces uvolňování dalších verzí ze strany Sun Microsystems. Kromě toho tato společnost také spravovala opravy chyb a rozšíření připravená komunitou. V současné době již tento projekt není dále aktivní.
- **Drizzle**[1]– fork MySQL připravený Brianem Akerem, jehož cílem je nabídnout rychlejší variantu MySQL obsahující pouze nejdůležitější a nejzákladnější funkce. Tento fork je určený především pro podporu webových aplikací. Základní funkcionalita je zajišťována jádrem, zatímco doplňkové funkce jsou zajišťovány zásuvnými moduly. Projekt sice ještě nezanikl, nicméně se zdá, že vývoj se na nějakou dobu pozastavil.
- **Percona Server**[2]– fork MySQL podporovaný konzultační společností Percona LLC. Cílem tohoto forku je vytvořit rychlou náhradu MySQL nabízející zvýšený výkon a některé další funkce, které nejsou součástí komunitní edice zveřejněné společností Oracle.
- **MariaDB**[3]– fork MySQL, který vytvořil sám Monty Widenius jako reakci na to, že společnost MySQL AB byla nejprve prodána společnosti Sun Microsystems a posléze se s i s ní stala součástí Oracle. Tento fork chce být především komunitní náhradou zajišťující rovnocenné funkce ve většině případů.

Další informace

Jednotlivé forky MySQL jsou podrobněji rozebrány v přednášce „Different MySQL Forks for Different Folks"[4], kterou přednesla Sheeri Cabral na konferenci Confoo v roce 2013.

1 *http://www.drizzle.org/*

2 *https://www.percona.com/software/mysql-database/percona-server*

3 *https://mariadb.org/*

4 *https://www.youtube.com/watch?v=dcWoHusSAsE*

I přes určité napětí v komunitě je dlouhodobý výhled pro „značku“ MySQL dobrý. Na rozdíl od očekávání mnoha lidí společnost Oracle další vývoj databázového systému MySQL neukončila. Dokonce lze říci, že kvalita jednotlivých vydání se pod vedením společnosti Oracle ve skutečnosti zvýšila. Jednotlivé forky tak zajišťují konkurenci, což může – doufejme – systému MySQL jedině prospět. I když se na celý vývoj podíváme tím nejpesimističtějším náhledem, zjistíme, že v poslední dekádě došlo v podnicích k nárůstu využívání otevřeného softwaru, díky čemuž nelze očekávat, že by se systém MySQL měl v nejbližší době stát pouhou historií.

Komu je tato kniha určena

Tato kniha je určena především pro ty z vás, kteří se zajímají o práci s daty a chtějí se naučit používat MySQL. A aby pro vás některé části této knihy byly skutečně přínosné, měli byste mít alespoň nějaké předcházející zkušenosti z oboru programování, byť znalost nějakého konkrétního jazyka není nutná.

Použité konvence

V této knize narazíte na řadu typografických konvencí a různých stylů rozvržení, které budou zvýrazňovat různé typy informací. Prohlédněte si proto následující konvence.

Ukázky zdrojového kódu

Zdrojový kód bude napsán neproporcionálním písmem, například takto:

```
<h1>Skvělý letní den</h1>
<p>Byl krásný letní den, který přímo vybízel na procházku parkem. Ptáčci
zpívali a všechny děti byly zpátky ve škole.</p>
```

Když budeme k současnému příkladu přidávat další zdrojový kód, takový kód se bude zobrazovat tučně:

```
function animuj() {
    nova_promenna = "Ahoj";
}
```

Místo již napsaného zdrojového kódu bude výpis obvykle obsahovat tři tečky:

```
function animuj() {
    ...
    return nova_promenna;
}
```

Tipy, poznámky a upozornění

Tip: Haló, vy tam!

Tipy vám poskytují malé cenné rady.

Poznámka: Ehm, promiňte...

Poznámky jsou doplňující informace, které se týkají tématu, ale nejsou kriticky důležité.

Upozornění: Nezapomeňte vždy...

... věnovat pozornost těmto sdělením.

Zpětná vazba od čtenářů

Nakladatelství a vydavatelství Computer Press, které pro vás tuto knihu přeložilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

Computer Press
Albatros Media a.s., pobočka Brno
IBC
Příkop 4
602 00 Brno

nebo

sefredaktor.pc@albatrosmedia.cz

Computer Press neposkytuje rady ani jakýkoli servis pro aplikace třetích stran. Pokud budete mít dotaz k programu, obraťte se prosím na jeho tvůrce.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nelze. Pokud v některé z našich knih nějakou najdete, ať už v textu nebo v kódu, budeme rádi, pokud nám ji oznámíte.

Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/K2274* po klepnutí na odkaz Soubory ke stažení. (Nejsou-li žádná errata zatím k dispozici, není odkaz Soubory ke stažení dostupný.)

KAPITOLA 1

Začínáme pracovat s MySQL

V této kapitole:

- Instalace MySQL na Linuxu
- Instalace MySQL ve Windows
- Komunikace se serverem
- Účty a zabezpečení MySQL
- Shrnutí

V této kapitole se seznámíte s prvními kroky, které budete muset provést, budete-li chtít začít používat MySQL. Ukážeme si, jak se MySQL instaluje jak na platformě Linuxu, tak i na platformě Windows. Proto při čtení textu dávejte pozor na to, abyste postupovali podle informací vztahujících se k vámi zvolené platformě. Poté uvidíte, jak se pracuje s klientem příkazového řádku systému MySQL, neboť právě tohoto klienta využijeme pro připojení k databázovému serveru a vytvoření naší první databáze.

Prvním krokem při instalování mnoha aplikací velice často bývá určení vhodné verze. Z tohoto důvodu je nyní vhodné připomenout, že systém MySQL je dostupný v několika edicích či variantách. Společnost Oracle vám nabízí jak volně dostupnou edici Community Edition, tak také placené edice s názvy Standard, Enteprise a Cluster Carrier Grade. Rozdíly mezi Community Edition a placenými edicemi spočívají především v licencování a podpoře možnosti využívání některých dodatečných serverových zásuvných modulů a nástrojů pro zálohování a monitorování.

Jak jsme však již řekli, MySQL je volně šiřitelný software, publikovaný pod licencí GNU General Public License, a proto by vás nemělo překvapit to, že existují i alternativní forky. Dvěma oblíbenými forky jsou MariaDB, což je komunitou udržovaná „rozšířená náhrada" MySQL, a Percona Server, podporovaný konzultační společností Percona LLC. Běžný uživatel si však rozdílů mezi MySQL, MariaDB a Percona Server většinou ani nevšimne.

Samozřejmě můžete použít kteroukoliv ze zmíněných verzí či edicí MySQL, nicméně v rámci snahy o udržení pozornosti a konzistence budu já osobně pracovat s edicí MySQL Community Edition verze 5.7.11 (stabilní verze, aktuální v době překladu této knihy). Současně budu veškeré dále popisované postupy omezovat na operační systémy Debian/Ubuntu, RedHat/CentOS a Windows Server 2012. Domnívám se totiž, že toto jsou ty

operační systémy, na nichž budete nasazovat MySQL v produktivním prostředí s největší pravděpodobností.

Poznámka: Lokální vývojové prostředí

Těm z vás, kteří si chtějí připravit instalaci MySQL pro účely lokálního vývoje a testování, doporučuji vytvoření virtuálního stroje pomocí nástroje VirtualBox[1] společnosti Oracle. V tomto virtuálním stroji si můžete nainstalovat jeden z výše uvedených operačních systémů a následně – podle instrukcí, které najdete níže – si nainstalovat i MySQL. Toto řešení vám dává možnost nejenom pracovat ve vývojovém prostředí, jehož konfigurace se bude v maximální možné míře podobat konfiguraci produktivního prostředí, aniž by bylo vázáno na určitý server či určitou síť, ale navíc i váš lokální systém bude ušetřen instalace dalších služeb a aplikací, a to bez ohledu na to, zda tím lokálním systémem je Linux, Windows či Mac OS X.

Instalace MySQL na Linuxu

Operační systém Linux rozhodně nepředstavuje homogenní platformu. Mimo jiné i proto, že jednotlivé distribuce se liší i preferovaným způsobem instalace softwaru. V této části si ukážeme, jak se MySQL instaluje na systémech Debian/Ubuntu a Red Hat/CentOS pomocí správce balíčků a jak se dá MySQL zkompilovat a následně i nainstalovat ze zdrojových kódů. Získáte tak dovednosti nezbytné k provedení instalace MySQL na systém Linux téměř ve všech scénářích, se kterými se můžete setkat.

Instalace pomocí správce balíčků

Většina moderních systémů řady Linux využívá ke značnému zjednodušení instalací dalších softwarů nějakého správce balíčků. A jelikož je databázový systém MySQL natolik rozšířený, je velice pravděpodobné, že buď přímo MySQL, nebo některý z jeho forků již existuje v repozitáři balíčků vámi zvolené distribuce. Pokud je nám známo, například Debian/Ubuntu nabízí ve svých repozitářích edici MySQL Community Edition, takže uživatelé mohou MySQL nainstalovat zadáním jednoduchého příkazu `sudo apt-get install mysql-server`. Naopak v repozitářích Red Hat/CentOS byl systém MySQL nedávno nahrazen forkem MariaDB; v tomto případě je tedy nutné k instalaci použít příkaz `su –c 'yum install mariadb-server'`.

Instalace softwaru z repozitáře udržovaného tvůrci distribuce je pro většinu uživatelů optimální, nicméně je nutné si uvědomit, že budete-li spoléhat na tyto repozitáře, může se stát, že výsledkem instalace nebude nejnovější verze. Naštěstí se ale nemusíme vzdát pohodlí, které nám práce s balíčky nabízí. Společnost Oracle totiž udržuje aktuální balíčky ve formátech RPM i DEB, které si můžete nainstalovat pomocí příkazů `rpm` a `dpkg`. Kromě toho ale

1 *https://www.virtualbox.org/*

společnost Oracle udržuje i repozitáře APT a Yum a nabízí speciální balíčky automaticky přidávající tyto repozitáře do seznamu známých repozitářů vašeho systému.

Pomocí následujících kroků můžete ve svém systému zaregistrovat jeden z repozitářů společnosti Oracle a následně z něj nainstalovat MySQL Community Edition. Neběží-li na vašem serveru grafické uživatelské rozhraní a nemůžete-li použít nějaký textový webový prohlížeč, například Lynx, budete muset provést první čtyři kroky na nějakém jiném systému a nakopírovat takto získaný soubor na váš server.

1. Spusťte si webový prohlížeč a přejděte na stránku repozitářů MySQL, mající adresu *http://dev.mysql.com/downloads/repo*.
2. V závislosti na správci balíčků vámi použité distribuce Linuxu klepněte na odkaz **DOWNLOAD** nacházející se v části **MySQL Yum Repository** anebo **MySQL APT Repository**. Budete přesměrováni na stránku obsahující seznam různých konfiguračních balíčků.
3. Dále klepněte na tlačítko **Download** nacházející se vpravo od balíčku vhodného pro váš systém. Například uživatelé systému Red Hat/CentOS 7 by si měli stáhnout balíček s názvem **Red Hat Enterprise Linux 7 / Oracle Linux 7 (Architecture Independent), RPM Package**. Naopak uživatelé systému Ubuntu 14.04 Trusty Tahr by si měli stáhnout balíček s názvem **Ubuntu / Debian (Architecture Independent), DEB**.
4. V dalším kroku se vás společnost Oracle bude snažit přesvědčit k tomu, abyste si u ní vytvořili nějaký účet. Tento krok však není povinný, a proto se přesuňte do spodní části stránky, kde klepněte na odkaz **No thanks, just start my download**. Poté by se již mělo zahájit stahování vámi zvoleného balíčku.
5. Otevřete si okno terminálu, přejděte do složky, do níž jste stáhli (či nakopírovali) zvolený balíček, a spuštěním odpovídajícího příkazu tento balíček nainstalujte:
   - Uživatelé systému Red Hat/CentOS by k instalaci měli použít příkaz:
     ```
     rpm –i mysql57-community-release-el7-7.noarch.rpm
     ```
   - Uživatelé systému Debian/Ubuntu by k instalaci měli použít příkaz:
     ```
     dpkg –i mysql-apt-config_0.6.0-1_all.deb
     ```
6. Repozitář je nyní zaregistrován a vy již můžete nainstalovat MySQL Community Edition pomocí správce balíčků své distribuce:
   - Uživatelé systému Red Hat/CentOS by k instalaci měli použít příkaz:
     ```
     su –c 'yum install mysql-community-server'
     ```
   - Uživatelé systému Debian/Ubuntu by k instalaci měli použít příkaz:
     ```
     sudo apt-get install mysql-server-5.7
     ```

Uživatelé systému Ubuntu budou v průběhu instalace navíc vyzváni k zadání hesla uživatele root systému MySQL (uživatelé systémů Debian a Red Hat/CentOS budou moci zadat toto heslo až v následujícím kroku, po spuštění jednoho poinstalačního příkazu). V této souvislosti je nutné zdůraznit, že MySQL si udržuje svůj vlastní seznam uživatelských účtů,

které jsou nezávislé na uživatelských účtech operačního systému – jinými slovy řečeno, byť uživatelské jméno může být shodné, uživatel root databázového systému MySQL není shodný s uživatelem root Linuxu.

Uživatelé systému Red Hat/CentOS by poté měli spustit tyto poinstalační příkazy, pomocí nichž nastaví heslo uživatele root MySQL, zaregistrují MySQL jako systémovou službu a spustí její instanci (systémy Debian/Ubuntu automaticky registrují a spouštějí MySQL):

1. K nastavení hesla uživatele root databázového systému MySQL použijte příkaz:
   `mysqladmin –u root heslo`
2. Spouštění MySQL během startu operačního systému zajistěte příkazem:
   `su –c 'chkconfig –level 2345 mysqld on'`
3. A nakonec spusťte server MySQL příkazem:
   `su –c 'systemctl start mysql'`

Tímto je dokončena instalace MySQL Community Edition na váš systém. Pro vaši budoucí práci níže uvádíme příkazy sloužící ke spuštění serveru MySQL, jeho zastavení a kontrole jeho stavu:

- Spuštění MySQL
  - Ubuntu – `sudo service mysql start`
  - Debian – `sudo systemctl start mysqld`
  - Red Hat/CentOS – `su –c 'systemctl start mysql'`
- Zastavení MySQL
  - Ubuntu – `sudo service mysql stop`
  - Debian – `sudo systemctl stop mysqld`
  - Red Hat/CentOS – `su –c 'systemctl stop mysql'`
- Zjištění aktuálního stavu MySQL
  - Ubuntu – `service mysql status`
  - Debian – `sudo systemctl status mysqld`
  - Red Hat/CentOS – `su –c 'systemctl status mysql'`

Poznámka: Jednodušší budoucnost

Ke spouštění, zastavování a kontrole stavu MySQL se používají rozdílné příkazy proto, že Ubuntu 14.04 používá Upstart, zatímco ostatní popisované distribuce používají systemd. Vývojáři Ubuntu předpokládají, že přechod k init systému systemd zahájí od verze 15.04. V době, kdy bude dostupná verze 16.04 s dlouhodobou podporou, by již příkazy pro provádění těchto úkonů měly být zcela shodné s těmi, které se používají v distribuci Debian.

Instalace ze zdrojů

Systémoví administrátoři kompilují software z jeho zdrojových kódů stále méně často, nicméně taková kompilace vám nabídne možnost úplné kontroly funkcí aplikace, její optimalizace a kontroly jejích konfiguračních nastavení. Jak však asi tušíte, jedná se o nejnáročnější metodu instalace.

V následujícím postupu jsou popsány kroky, které musíte provést, chcete-li si stáhnout zdrojový kód serveru MySQL Community Edition, zkompilovat jej a nakonec nainstalovat. Opět platí, že nemáte-li přístup ke grafickému uživatelskému rozhraní či textovému webovému prohlížeči přímo na serveru, budete muset provést prvních několik kroků na jiném počítači a stažené soubory přenést na požadovaný počítač.

1. Spusťte si webový prohlížeč a přejděte na stránku MySQL Community Downloads, mající adresu *http://dev.mysql.com/downloads*.
2. Klepnutím na odkaz **MySQL Community Server** přejděte na stránku Download MySQL Community Server. Pro výběr požadované cílové platformy můžete využít rozbalovací seznam **Select Platform**.
3. V tomto seznamu si zvolte **Source Code**, poté se na stránce přesuňte dolů k položce **Generic Linux (Architecture Independent), Compressed TAR Archive** a vpravo vedle ní klepněte na odkaz **Download**.
4. Vytvoření účtu není pro pokračování ve stahování nezbytné. Proto přejděte do spodní části stránky a zde klepněte na odkaz **No thanks, just start my download**.
5. Otevřete si okno terminálu a pomocí následujících příkazů vytvořte uživatelský účet, určený pouze ke spouštění serveru MySQL:

```
sudo groupadd mysql
sudo useradd –r –g mysql mysql
```

6. Poté přejděte do složky, do níž jste stáhli zkomprimovaný zdrojový kód. Tento archiv rozbalte a poté přejděte do složky s kódem:

```
cd /tmp
gzip –cd mysql-5.7.11.tar.gz | tar xvf –
cd mysql-5.7.11
```

7. Spuštěním příkazu `cmake` vytvořte skripty pro sestavení. V následující ukázce tohoto příkazu neuvádím žádné volby, nicméně úplný přehled konfiguračních voleb najdete v online dokumentaci[2].

```
cmake
```

8. Poté spusťte příkaz `make`, jímž server MySQL zkompilujete. Následuje spuštění příkazu `make install`, vyžadující nejvyšší oprávnění. Díky tomu budou výsledné

2 *http://dev.mysql.com/doc/refman/5.7/en/source-configuration-options.html*

binární soubory, pomocné nástroje, knihovny a soubory dokumentace překopírovány do nové domovské složky nacházející se ve vašem systému:

```
make
sudo make install
```

9. Níže vidíte příkazy, jimiž můžete zajistit to, že nainstalované soubory budou mít přiřazeného správného vlastníka a budou k nim nastavena odpovídající přístupová oprávnění:

```
sudo chown –R mysql /usr/local/mysql
sudo chgrp –R mysql /usr/local/mysql
```

10. Datová složka serveru MySQL a jeho systémové tabulky musí být inicializovány skriptem `mysql_install_db`, který najdete v podsložce `scripts` instalační složky. Skript pracuje s cestami zadávanými relativně vůči instalační složce, a proto tento skript raději spouštějte z instalační složky, a nikoliv z podsložky `scripts` či odkudkoliv jinud:

```
cd /usr/local/mysql
sudo scripts/mysql_install_db –user=mysql
```

11. Spusťte server MySQL a nastavte heslo jeho uživatele root:

```
sudo mysqld_safe &
mysqladmin –u root heslo
```

Tímto je celá instalace serveru MySQL dokončena. Existují však ještě další úkony, související s konfigurací systému, které byste měli zvážit. Doporučuji vám, abyste přidali podsložku `bin` instalační složky do proměnné prostředí `PATH`. Díky tomu budete moci spouštět pomocné nástroje serveru MySQL, aniž byste vždy museli zadávat plnou cestu. Za předpokladu, že používáte Bash, můžete do konfiguračního souboru `/etc/profile` přidat následující řádky:

```
PATH=/usr/local/mysql/bin:$PATH
export PATH
```

Poznámka: Práce s proměnnou prostředí PATH

Nastavením hodnoty proměnné prostředí `PATH` v souboru `/etc/profile` zjednodušíte přístup k pomocným nástrojům MySQL všem uživatelům systému. Pokud ovšem chcete, abyste takto jednoduchý přístup ke zmiňovaným nástrojům měli pouze vy, přidejte výše uvedené řádky do souboru `~/.bash_profile` či `~/.bashrc`.

Je také velice pravděpodobné, že budete chtít zajistit automatické spouštění MySQL v době náběhu systému. Níže uvedené kroky vycházejí z předpokladu, že používáte Linux založený na init systému System V.

1. Přejděte do podsložky `support-files` zdrojového kódu. V ní byste měli najít soubor `mysql.server`, který je nutné zkopírovat do složky `init.d` vašeho systému a poté jej změnit na spustitelný:

```
sudo cp /tmp/mysql-5.7.11/support-files/mysql.server /etc/init.d/mysql
sudo chmod 755 /etc/init.d/mysql
```

2. V jednotlivých požadovaných úrovních běhu vytvořte symbolické odkazy ukazující na tento skript:

```
ln –s /etc/init.d/mysql /etc/rc3.d/S99mysql
ln –s /etc/init.d/mysql /etc/rc0.d/K01mysql
```

A nyní již můžete spustit server MySQL příkazem `sudo /etc/init.d/mysql start`, zatímco k jeho zastavení můžete použít příkaz `sudo /etc/init.d/mysql stop`.

Instalace MySQL ve Windows

I přesto, že společnost Microsoft vždy aktivně udržuje a podporuje několik verzí operačního systému Windows najednou, lze říci, že ve srovnání se systémy Linux jsou operační systémy řady Windows poměrně homogenní. Níže uvedené kroky se týkají systému Windows Server 2012, nicméně lze předpokládat, že ve větší či menší míře budou použitelné i pro desktopový systém Windows 8 či Windows 8.1.

1. Spusťte si webový prohlížeč a přejděte na stránku MySQL Community Downloads, mající adresu *http://dev.mysql.com/downloads*.
2. Klepnutím na odkaz **MySQL on Windows (Installer & Tools)** přejděte na stránku MySQL on Windows. Na ní klepněte na první odkaz, jímž je **MySQL Installer**.
3. Dostanete se tak na stránku Download MySQL Installer. Přesuňte se dolů k položce **Windows (x86, 32-bit), MSI Installer** a vpravo vedle ní klepněte na odkaz **Download**.
4. V dalším kroku se vás společnost Oracle bude snažit přesvědčit k tomu, abyste si u ní vytvořili nějaký účet. Tento krok však není povinný, a proto se přesuňte do spodní části stránky, kde klepněte na odkaz **No thanks, just start my download**. Poté by se již mělo zahájit stahování vámi zvoleného balíčku.
5. Přejděte do složky, do níž jste si uložili stažený soubor ve formátu MSI, a poklepáním na něj spusťte průvodce instalací.
6. Na obrazovce License Agreement zaškrtněte volbu, jíž potvrdíte souhlas s licenčními podmínkami, a poté klepněte na tlačítko **Next**.
7. Dostanete se tak na obrazovku Choose Setup Type, na níž zvolte **Server Only**. Následně klepnutím na tlačítko **Execute** zahajte samotnou instalaci. V závislosti na nastavených politikách zabezpečení se může stát, že se vám zobrazí dialog User Account Control.
8. Po dokončení instalace klepněte na tlačítko **Next**, čímž přejdete ke konfiguraci MySQL.

9. Prvním krokem je nastavení typu konfigurace. Doporučujeme vám vybrat volbu **Development Machine**, neboť díky ní by MySQL server měl využívat pouze minimální množství systémových prostředků.
10. Pokud se týká ostatních voleb tohoto dialogu (použití portu 3306 protokolu TCP/IP), ponechte je ve výchozím nastavení a pokračujte klepnutím na tlačítko **Next**.
11. V dalším kroku budete požádáni o zadání hesla uživatele root. Zadejte tedy heslo a ve spodní části dialogu opět klepněte na tlačítko **Next**.
12. Volby, které se vám zobrazí nyní, se týkají spouštění serveru MySQL jako služby systému Windows. Ve většině případů asi budete chtít, aby server MySQL byl nakonfigurován jako služba systému Windows a aby se tato služba spouštěla spolu s operačním systémem. Ve spodní části dialogu pak můžete nastavit účet, pod nímž bude tato služba spouštěna. Opět lze říci, že ve většině případů bude vyhovovat výchozí nastavení, tj. spouštění služby pod identitou uživatele `SYSTEM`. Pokračujte klepnutím na tlačítko **Next**.
13. Dostáváte se k poslednímu kroku, jímž je provedení vlastní konfigurace. Nejprve se vám zobrazí seznam všech potřebných úkonů. Celý proces spusťte klepnutím na tlačítko **Execute**.
14. Jsou-li všechny úkony dokončeny úspěšně, u každého z nich se zobrazí zelená ikona zaškrtnutí a ve spodní části dialogu se zpřístupní tlačítko **Finish**. Klepnutím na něj se vrátíte do průvodce instalací, v němž rovnou klepněte na tlačítko **Next**.
15. Dostanete se tak na poslední obrazovku průvodce instalací, která vám nabídne pouze možnost zkopírování protokolu celé instalace do schránky. Celý proces ukončete klepnutím na tlačítko **Finish**.

Nyní podle následujícího postupu přidejte podsložku `bin` té složky, do níž byl instalován server MySQL, do systémové proměnné `PATH`.

1. Otevřete si okno **System Properties**.
   a. Stisknutím klávesové kombinace **Win+C** si zobrazte uživatelské rozhraní Edge.
   b. Klepněte na charm **Search**, zadejte Control Panel, a jakmile se ve výsledcích objeví ikona Control Panel, klepněte na ni.
   c. Zobrazí-li se vám ovládací panely pomocí pohledu **Category**, klepněte na položku **System and Security** a následně na položku **System**. Jestliže se vám ovládací panely zobrazí v pohledu **Icon**, klepněte na ikonu **System**.
2. V levé části dialogu klepněte na odkaz **Advanced system settings**, jímž si otevřete okno **System Properties**.
3. Pokud se vám toto okno neotevřelo rovnou na záložce **Advanced**, přejděte na ni a v její spodní části pak klepněte na tlačítko **Environment Variables**.
4. V sekci **System variables** si vyberte proměnnou **Path** a klepněte na tlačítko **Edit**.
5. Na konec stávající hodnoty přidejte cestu ke složce `bin` (`C:\Program Files\MySQL\MySQL Server 5.7\bin`), přičemž tuto novou položku oddělte od hodnoty stávající středníkem.

6. Klepnutím na tlačítko **OK** postupně všechny dialogy uzavřete.

V tuto chvíli jste tedy schopni spouštět jednotlivé pomocné nástroje z okna příkazového řádku, aniž byste přitom museli zadávat úplnou cestu: složka `bin` je totiž součástí toho seznamu cest, v němž systém Windows vyhledává spustitelné soubory. A jelikož byl server MySQL již během instalace zaregistrován jako služba systému Windows, bude se spouštět automaticky během náběhu systému a budete jej moci řídit pomocí správce služeb systému Windows. V případě potřeby si však můžete otevřít okno příkazového řádku – pochopitelně s oprávněními administrátora – a server MySQL spustit, příp. zastavit, pomocí následujících příkazů:

- Spuštění serveru MySQL – `net start MySQL57`
- Zastavení serveru MySQL – `net stop MySQL57`

Komunikace se serverem

Na první pohled se vám může zdát, že server MySQL vlastně jen nečinně běží a čeká na to, až obdrží nějaké dotazy. Jakmile server nějaký dotaz přijme, provede jej za nás a odešle zpět výsledek. Přitom existuje několik způsobů, jimiž můžeme se serverem MySQL komunikovat: například s ním můžeme komunikovat programově z prostředí aplikace, kterou jsme si sami napsali, anebo s ním můžeme komunikovat interaktivně pomocí vyhrazeného klientského programu. Ve zbývajících částech této knihy budeme ke spojení a komunikaci používat převážně klienta příkazového řádku, který je součástí standardní instalace serveru MySQL; výjimkou bude kapitola 5, v níž se budeme bavit o odesílání příkazů jazyka SQL z prostředí nějakého programu.

Otevřete si okno příkazového řádku (či okno terminálu) a zadejte do něj příkaz `mysql -u root -p`. Volba `-u` určuje uživatelský účet serveru MySQL, který bude použit pro připojení, zatímco díky volbě `-p` se zobrazí výzva pro zadání hesla tohoto uživatele. Jakmile vás o to program požádá, zadejte heslo uživatele root, které jste zadali v předcházejících částech této kapitoly.

Poznámka: Mnoho voleb

Volby **`-u`** a **`-p`** jsou pouze dvěma z mnoha voleb, s nimiž je klientský program schopen pracovat. Níže najdete přehled některých dalších voleb, které možná začnete využívat poměrně často (chcete-li si zobrazit úplný přehled všech voleb, spusťte klienta s volbou -?):

- **`-A`** – vypnutí automatického dokončování názvů
- **`-B`** – běh v dávkovém režimu (při zobrazování výsledků se jako oddělovač využívá znak tabelátoru)
- **`-e`** *`příkaz`* – spuštění zadaného příkazu jazyka SQL
- **`-h`** *`název_hostitele`* – zadání hostitelského názvu vzdáleného databázového serveru
- **`-N`** – potlačení zobrazování názvů sloupců ve výstupu

- `-p` – zobrazení výzvy pro zadání hesla toho uživatelského účtu, který má být použit k připojení
- `-u` *`jméno_uživatele`* – umožňuje zadání jména toho uživatele, který má být použit k připojení
- `-?` – zobrazení seznamu všech podporovaných voleb

Jakmile se úspěšně připojíte k serveru MySQL, zobrazí se v okně příkazového řádku výzva `mysql>`. A právě z této výzvy budeme zadávat všechny příkazy jazyka SQL. Po zadání nějakého příkazu klient zobrazí nejen samotnou odpověď serveru, ale i informace o době, kterou ke zpracování dotazu server potřeboval, a o veškerých chybách, na něž během provádění dotazu narazil.

Server MySQL je samozřejmě schopen udržovat najednou více databází. Chcete-li zjistit, které databáze daný server spravuje, zadejte na řádku výzvy příkaz `SHOW DATABASES;`. Výsledkem takového dotazu bude seznam všech databází aktuálně spravovaných daným serverem. Připojíte-li se k nově nainstalované instanci, zobrazí se vám pouze čtyři databáze využívané samotným serverem MySQL: `information_schema`, `mysql`, `performance_schema` a `sys`. V některých případech se můžete setkat i s databází `test`, vznikající po spuštění nástroje `mysql_install_db` a určené pro testování.

Chcete-li vytvořit novou databázi, použijte příkaz `CREATE DATABASE`. Například je-li vaším cílem vytvoření databáze s názvem „mysql_okamzite“, zadejte na řádku výzvy příkaz `CREATE DATABASE mysql_okamzite;`. Poté znovu proveďte příkaz `SHOW DATABASES;` a ve výstupu již uvidíte svoji první databázi.

Potřebujete-li serveru říci, že chcete pracovat s určitou databází, použijte příkaz `USE`. Zadáte-li na řádku výzvy příkaz `USE mysql_okamzite;`, budou všechny následně zadané příkazy prováděny v databázi `mysql_okamzite`. Cílovou databázi je možné zadat i ve chvíli, kdy se k serveru připojujete pomocí klienta příkazového řádku. Typickou ukázkou by mohl být příkaz `mysql –u root –p mysql_okamzite`.

Příkaz `SHOW TABLES` serveru MySQL říká, že nám má zobrazit seznam všech tabulek právě aktivní databáze (tj. té, k níž jsme se připojili pomocí příkazu `USE`). Je zřejmé, že jelikož jsme v databázi `mysql_okamzite` nevytvořili žádné tabulky, bude odpověď na příkaz `SHOW TABLES` poměrně jednoduchá: „Empty set“. Vytvoření správné tabulky však vyžaduje poněkud delší plánování a přípravu a navíc jsme již probrali poměrně dost nových témat. Z těchto důvodů se tematice vytváření nových tabulek budeme věnovat až v následující kapitole.

Chcete-li ukončit připojení k serveru MySQL pomocí klienta příkazového řádku, jednoduše buď zadejte příkaz `exit`, anebo stiskněte klávesovou kombinaci **Ctrl+D**.

Účty a zabezpečení MySQL

Posledním tématem, jemuž bychom se měli v této kapitole věnovat, jsou uživatelské účty serveru MySQL. Byť uživatel root serveru MySQL není totožný se stejnojmenným uživatelem operačního systému vašeho počítače, rozhodně není určen k běžnému užívání. Uživatel root serveru MySQL by měl být používán pouze k účelům správy serveru, například při vytváření dalších uživatelských účtů, při nastavování oprávnění či při čistění mezipaměti přístupu. Pro každodenní práci by rozhodně měly být využívány účty s nižšími oprávněními.

Chcete-li vytvořit nový uživatelský účet, pomocí klienta příkazového řádku se připojte k serveru MySQL, a to jako uživatel root. Poté proveďte následující příkaz `CREATE USER`:

```
CREATE USER 'cpress'@'localhost' IDENTIFIED BY 'tajne';
```

Tento příkaz vytvoří nový uživatelský účet s názvem „cpress“ a heslem „tajne“, umožňující uživateli ověření z téhož systému, na němž běží server MySQL. Namísto localhost můžete použít názvy jiných počítačů či jejich IP adresy, čímž umožníte připojení z rozdílných systémů a sítí. Mějte však na mysli, že server MySQL považuje každou kombinaci jméno uživatele / název hostitelského počítače za samostatný účet. To ovšem znamená, že cpress@localhost a cpress@192.168.1.100 jsou ve skutečnosti považovány za dva nezávislé a samostatné účty, z nichž každý má přiřazenou svoji vlastní sadu oprávnění.

Poznámka: Zástupné znaky

Znaky _ a % jsou zástupnými znaky, které můžete použít v části výše uvedeného příkazu pro identifikaci hostitelského počítače, a to za účelem zajištění částečných shod. Například tedy můžete použít definice typu „192.168.1.10_“ anebo „%.example.com“. Přitom znak _ nahrazuje jeden libovolný znak, zatímco znak % nahrazuje libovolný počet znaků. Z toho vyplývá, že následující příkaz můžete využít k vytvoření účtu, který je schopen se ověřit odkudkoliv, z kteréhokoliv hostitelského systému – což je sice velmi pohodlná, avšak na druhou stranu velice nebezpečná praxe:
`CREATE USER 'cpress'@'%' IDENTIFIED BY 'tajne';`

To, zda server MySQL umožní určitému uživateli provedení nějakého úkonu, závisí na oprávněních přiřazených jeho uživatelskému účtu. Nové účty jsou vytvářeny bez jakýchkoliv oprávnění, což znamená, že jim musíme explicitně přiřadit všechna oprávnění, která budou potřebovat. Uživatel „cpress“ bude potřebovat několik oprávnění, neboť jej budete využívat i v dalších částech této knihy. V tuto chvíli stačí, když tomuto účtu přiřadíme základní sadu oprávnění (další oprávnění pak můžete přiřadit ve chvíli, kdy začnou být nezbytná). Zadejte následující příkaz:

```
GRANT CREATE, DROP, ALTER, INSERT, UPDATE, SELECT, DELETE, INDEX, REFERENCES ON
mysql_okamzite.* TO 'cpress'@'localhost';
```

Syntaxe příkazu `GRANT` serveru MySQL je natolik flexibilní, že rozsah oprávnění můžeme zúžit pouze na vybrané sloupce určité tabulky či na vybrané tabulky databáze. V našem

případě jsme však serveru MySQL řekli, aby uživateli „cpress“ přihlašujícímu se z počítače „localhost“ přiřadil veškerá oprávnění ke všem tabulkám databáze `mysql_okamzite` (což je dáno znakem *). Přiřazená oprávnění jsou:

- `CREATE` – umožňuje uživateli vytváření databází a tabulek
- `DROP` – umožňuje uživateli výmaz celých tabulek a databází
- `ALTER` – umožňuje uživateli změnu definice existující tabulky
- `INSERT` – umožňuje uživateli přidávání nových záznamů do již existující tabulky
- `UPDATE` – umožňuje uživateli aktualizaci záznamů v tabulce
- `SELECT` – umožňuje uživateli načítání již existujících záznamů z tabulky
- `DELETE` – umožňuje uživateli výmaz existujících záznamů z tabulky
- `INDEX` – umožňuje uživateli vytváření či výmaz indexů

Přehled všech možných oprávnění a toho, co vlastně uživatelskému účtu umožňují, najdete v dokumentaci[3]. Jak jsme si řekli již výše, pokud v budoucnu zjistíte, že určitý účet vyžaduje nějaká nová oprávnění, stačí provést další příkaz `GRANT`. Obdobně platí, že oprávnění, která nejsou nadále potřebná, lze uživatelskému účtu odebrat; k těmto účelům je však nutné použít jiný příkaz, a to příkaz `REVOKE`, jehož syntaxe je velice podobná syntaxi příkazu `GRANT`:

```
REVOKE CREATE, DROP, ALTER, INDEX ON mysql_okamzite.* TO 'cpress'@'localhost';
```

Kdykoliv však provádíte nějakou změnu týkající se uživatelského účtu či jeho oprávnění, musíte vzápětí spustit příkaz `FLUSH PRIVILEGES`, jímž serveru MySQL řeknete, aby si znovu načetl mezipaměť informací o účtech, kterou si udržuje. Jedině tak zajistíte to, že změny vstoupí v platnost okamžitě. Neučiníte-li tak, provedené změny v podstatě nebudou mít žádný vliv, a to až do doby následného restartu serveru MySQL:

```
FLUSH PRIVILEGES;
```

Po provedení tohoto příkazu ukončete připojení k serveru MySQL z klienta příkazového řádku a poté se znovu připojte pomocí účtu „cpress“. Pokud jste zadali všechny výše uvedené příkazy správně a pokud po zobrazení výzvy zadáte i správné heslo, znovu se vám zobrazí výzva `mysql>`.

Shrnutí

V této kapitole jsme probrali spoustu témat. Naučili jste se instalovat MySQL na různé platformy, dozvěděli jste se, jak se můžete připojit k serveru MySQL pomocí klienta příkazového řádku, seznámili jste se s postupem vytváření nových databází a nakonec jste alespoň ve stručnosti viděli, jak se v prostředí serveru MySQL spravují uživatelé.

3 *http://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html*

Ačkoliv nyní možná chcete ihned pokračovat studiem další kapitoly, já osobně vám doporučuji, abyste si nejdříve alespoň stručně pročetli online manuál serveru MySQL – přinejmenším ty části, v nichž jsou popsána dosud probraná témata. Seznamte se s podrobnostmi týkajícími se příkazů `CREATE USER` a `GRANT`. Zjistěte si, jak se dá změnit heslo nějakého existujícího uživatelského účtu a jak se dá vymazat účet, který není nadále potřebný. Zamyslete se nad tím, jaká oprávnění byste přiřadili účtu používanému pro ukládání a načítání dat v rámci nějaké webové aplikace.

V kapitole 2 se budeme věnovat tématům souvisejícím s ukládáním dat v databázi. Uvidíte, jak se vytváří nová tabulka a jak se do ní vkládají nové řádky. Dozvíte se také, jaké typy dat lze do tabulek ukládat, co se míní pojmem databázové úložiště (storage engine) a jak naše volba databázového úložiště ovlivňuje způsob, jímž server MySQL spravuje naše data.

KAPITOLA 2

Ukládání dat

V této kapitole:

- Vytváření tabulek
- Přidávání dat
- Shrnutí

Data uložená v relační databázi jsou uspořádána do **tabulek**. Podíváme-li se pak na takovou tabulku, uvidíme, že data jsou v ní uložena do jakési mřížky, v níž každý **řádek** tvoří jeden záznam a každý **sloupec** identifikuje určitou hodnotu záznamu. Pro představu jsme níže připojili tabulku ukazující počet medailí, které získala každá z pěti medailově nejúspěšnějších zemí účastnících se zimních olympijských her v roce 2014. V každém řádku vidíte název země a dále počet zlatých, stříbrných a bronzových medailí, které vyhráli sportovci reprezentující danou zemi. V posledním sloupci pak vidíte celkový počet medailí.

Země	Zlato	Stříbro	Bronz	Celkem
Rusko	13	11	9	33
USA	9	7	12	28
Norsko	11	5	10	26
Kanada	10	10	5	25
Holandsko	8	7	9	24

Tabulka, kterou jsme si ukázali, je „fyzická", a to proto, že ji můžete vidět vytištěnou v knize či nakreslenou na tabuli. Taková tabulka je omezená pouze dostupným fyzickým prostorem.

Na druhou stranu databázová tabulka je nehmotnou strukturou uloženou kdesi na pevném disku či v paměti počítače. Můžeme si ji tedy pouze představit anebo si ji zkusit nakreslit. Databázová tabulka je interpretována počítačovým procesem (například serverem MySQL), přičemž omezení tohoto procesu definují i omezení týkající se tabulky. Počet sloupců, počet řádků, a dokonce i typy hodnot, které lze ukládat do jednotlivých sloupců to vše závisí na tom, co je schopen zpracovat nejenom použitý databázový server, ale i daný počítač. Avšak bez ohledu na tato omezení platí, že databázová tabulka je ve skutečnosti velice flexibilní. Můžeme vytvářet relace (tj. vazby) mezi jednotlivými tabulkami, můžeme kombinovat více tabulek dohromady, můžeme třídit záznamy a zobrazovat si pouze vybrané záznamy, můžeme odstraňovat záznamy a v neposlední řadě můžeme provádět i nejrůznější výpočty s daty uloženými v jednotlivých záznamech.

V této kapitole se zaměříme na příkaz `CREATE TABLE`, sloužící k vytváření nových databázových tabulek. V souvislosti s tím se budeme věnovat i některým důležitým detailům týkajícím se procesu vytváření tabulek: datovým typům podporovaným serverem MySQL, omezením názvů a databázovým úložištím (storage engine). Ukážeme si také, jak se pomocí příkazu `INSERT` vkládají do tabulky nové záznamy. Celou kapitolu pak zakončíme výkladem o transakcích.

Vytváření tabulek

Pro vytváření tabulek se používá příkaz `CREATE TABLE`. Ve své nejjednodušší podobě nám tento příkaz umožňuje zadání názvu tabulky, kterou chceme vytvořit, spolu se seznamem sloupců a jejich datových typů. Nemělo by vás však překvapit to, že v závislosti na požadavcích, na jejichž základě je tabulka vytvářena, se příkaz `CREATE TABLE` může stát velmi, velmi složitým. V rámci definice sloupce je totiž možné zadat jeden či více atributů; tyto atributy pak mohou omezovat rozsah hodnot, které je možné do sloupce ukládat, či definovat výchozí hodnotu používanou v případě, že uživatel nezadá žádnou vlastní hodnotu. Velice častá je také definice jakýchkoliv logických vazeb existujících mezi právě vytvářenou tabulkou a nějakou již existující tabulkou či určení toho databázového úložiště, které má být použito ke správě tabulky. Chcete-li se přesvědčit, jak podrobný a složitý příkaz `CREATE TABLE` může být, pak si prohlédněte jeho syntaxi a všechny volby v dokumentaci[1] serveru MySQL.

Podívejme se nyní na dvojici poměrně jednoduchých příkazů `CREATE TABLE`. (Sice se pokusím zdůraznit některé možnosti, které zvyšují složitost, ale slibuji, že to rozhodně nebudu příliš přehánět.) Přihlaste se k serveru MySQL a připojte se k databázi `mysql_okamzite`, kterou jste si připravili v kapitole 1. Poté zadejte níže uvedené příkazy. Přitom platí, že po každém z nich by server MySQL měl zareagovat hlášením „Query OK“.

```
CREATE TABLE zamestnanec (
    osobni_cislo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    prijmeni VARCHAR(30) NOT NULL,
    jmeno VARCHAR(30) NOT NULL,
    email VARCHAR(100) NOT NULL,
    datum_nastupu DATE NOT NULL,
    poznamka MEDIUMTEXT,

    PRIMARY KEY (osobni_cislo),
    INDEX (prijmeni),
    UNIQUE (email)
)
ENGINE=InnoDB;
```

1 *http://dev.mysql.com/doc/refman/5.7/en/create-table.html*

```
CREATE TABLE adresa (
    osobni_cislo INTEGER UNSIGNED NOT NULL,
    ulice VARCHAR(50) NOT NULL,
    mesto VARCHAR(30) NOT NULL,
    kraj CHAR(3) NOT NULL,
    psc CHAR(5) NOT NULL,

    FOREIGN KEY (osobni_cislo),
        REFERENCES zamestnanec (osobni_cislo)
)
ENGINE=InnoDB;
```

První příkaz slouží k vytvoření tabulky s názvem `zamestnanec`, určené pro ukládání základních informací o zaměstnancích společnosti – tj. pro ukládání jejich jmen a příjmení, emailových adres, dat nástupu a případně i nějakých poznámek, které by si k zaměstnanci chtěl uložit například vedoucí personálního odboru. Formátování, které vidíte, jsme použili jenom kvůli zvýšení čitelnosti kódu: samotnému serveru MySQL je totiž zcela jedno, jestli celý příkaz napíšeme na jeden jediný řádek anebo jej rozdělíme na několik řádků a použijeme odsazení. A při zadávání příkazu nezávisí ani na používání mezer.

Poznámka: Místní hledisko

Tabulka **`adresa`** je navržena tak, aby odpovídala běžným českým zvyklostem pro zápis adres s tím, že pro uložení PSČ je určeno pouze 5 znaků (bylo by tedy nutné zadávat PSČ bez mezer). Pokud byste však navrhovali takovou tabulku pro jinou zemi, museli byste zvážit i místní požadavky na zápis adresy v dané zemi – například v USA byste asi namísto kraje potřebovali ukládat informaci o státě, v Holandsku byste pro PSČ museli vyhradit 6 znaků apod.

Pro tabulku a její sloupce smíte použít v podstatě libovolné názvy vyhovující následujícím pravidlům:

- V názvech smí být použita jak malá, tak i velká písmena základní latinské abecedy (A–Z, a–z), znak dolaru ($), znak podtržítka (_) či znaky sady Unicode z rozmezí U+0080 – U+FFFF.
- V názvech není povoleno používání následujících znaků: znaku null (0x00), všech znaků sady Unicode v rozmezí U+10000 a vyšších, znaku tečka (.) a veškerých znaků, které nesmí být používány ani v názvech souborů, například dopředného lomítka (/) či zpětného lomítka (\).
- Obsahuje-li název nějaké znaky mimo výše specifikovaný rozsah, musí být uzavřeny do apostrofů. Server MySQL standardně používá pro tyto účely znak zpětného apostrofu (`…`), byť jej můžete nakonfigurovat i tak, aby používal znak normálního apostrofu ('…'). Doporučuji vám však, abyste se snažili využívat standardní nastavení.

- Použijete-li jako název nějaké rezervované slovo serveru MySQL, musíte jej uzavřít do zpětných apostrofů (případně jiných nakonfigurovaných znaků). Seznam rezervovaných slov najdete v online dokumentaci[2].

Sloupec `osobní_cislo` je navržen jako **primární klíč** (neboli primary key) tabulky. Primárním klíčem tabulky může být ten její sloupec, v němž jsou všechny hodnoty jedinečné, a lze je tedy využít k rychlé identifikaci jednotlivých záznamů tabulky. V případě návrhů složitějších tabulek je samozřejmě možné při definici primárního klíče použít více sloupců najednou, nicméně použití jednoho sloupce typu `INTEGER` je velice častým řešením. Přitom platí, že součástí definice jedné tabulky smí být pouze jeden primární klíč (a proto se tento klíč jmenuje *primární*).

Součástí definice sloupce `osobní_cislo` je však i atribut `AUTO_INCREMENT`. Kdykoliv se do tabulky pokusíme přidat záznam, který nebude obsahovat žádnou hodnotu pro tento sloupec, server MySQL automaticky použije to celé číslo, které následuje za poslední použitou hodnotou tohoto sloupce. Řekněme, že v tabulce `zamestnanec` již máme nějaký počet záznamů a že nejvyšší dosud použitá hodnota ve sloupci `osobní_cislo` je 42. Pokud bychom přidali nový záznam neobsahující žádný údaj pro sloupec `osobní_cislo`, server MySQL by automaticky použil číslo 43. A kdybychom přidali ještě další záznam neobsahující žádný údaj pro sloupec `osobní_cislo`, server MySQL by použil 44 atd. Přitom platí, že součástí definice jedné tabulky smí být pouze jeden sloupec označený jako sloupec s automatickým přírůstkem, a tento sloupec navíc musí být primárním klíčem.

Server MySQL v pozadí udržuje různé datové struktury sloužící ke sledování dat a jejich různých vazeb. Index, definovaný na sloupci `prijmeni`, serveru MySQL říká, že hodnoty z tohoto sloupce možná budeme chtít využívat jako výběrová kritéria při načítání dat – například ve chvíli, kdy budeme chtít vyhledat všechny zaměstnance s příjmením Novák či Nový. Server MySQL vytvoří a nadále bude udržovat speciální datovou strukturu obsahující hodnoty ze sloupce `prijmeni` a urychlující veškeré vyhledávání podle tohoto sloupce. V žádném případě byste to však s indexy neměli přehánět: server MySQL totiž k jejich údržbě potřebuje nějaký čas, takže i když načítání dat může být rychlejší, přidávání nových záznamů či aktualizace těch stávajících mohou být pomalejší.

Pojmem **omezení** (angl. constraint) se označuje speciální podmínka vztahující se na nějaký sloupec či celou tabulku. Přitom platí, že tato podmínka musí být splněna v každém časovém okamžiku. Většinu sloupců jsme definovali s omezením `NOT NULL`, což je omezení, které zabraňuje serveru MySQL v ukládání hodnoty typu `NULL` do těchto sloupců. Jinými slovy řečeno, omezení `NOT NULL` znamená, že sloupec musí obsahovat nějakou hodnotu. V této souvislosti je také nutné zdůraznit, že server MySQL rozlišuje hodnotu `NULL` a prázdnou hodnotu, například prázdný textový řetězec.

2 *http://dev.mysql.com/doc/refman/5.7/en/keywords.html*

Omezení UNIQUE, použité u sloupce email, zajišťuje to, že všechny emailové adresy uložené v tabulce budou jedinečné (budou se vzájemně lišit). Na první pohled se vám tedy může zdát, že omezení UNIQUE se podobá PRIMARY KEY, ale existují mezi nimi poměrně zásadní rozdíly. Jelikož hodnoty ve sloupci označeném jako primární klíč musí zajistit možnost jedinečné identifikace každého záznamu, je jejich jedinečnost v podstatě dána. V případě použití PRIMARY KEY tedy nemusíme explicitně specifikovat ještě UNIQUE. A zatímco jedna tabulka smí obsahovat pouze jeden sloupec definovaný jako primární klíč, součástí tabulky smí být libovolné množství sloupců definovaných s omezením UNIQUE. Kromě toho sloupec definovaný s omezením UNIQUE smí obsahovat i hodnotu NULL, což sloupec primárního klíče nesmí.

Omezení FOREIGN KEY, které vidíte v příkazu CREATE TABLE pro vytvoření tabulky adresa, se odkazuje na tabulku zamestnanec, čímž vlastně vytváří vazbu (relaci) mezi oběma tabulkami. Tato vazba znamená, že určitý řádek tabulky adresa je logicky propojen s nějakým řádkem tabulky zamestnanec majícím stejnou hodnotu ve sloupci osobni_cislo. Představme si například ten záznam tabulky adresa, jehož sloupec osobni_cislo obsahuje hodnotu 42. Tento řádek může být (a také je) propojen s tím řádkem tabulky zamestnanec, který ve sloupci osobni_cislo také obsahuje hodnotu 42. Jinými slovy řečeno, adresa osobního čísla 42 je propojena s osobními daty osobního čísla 42. Sloupec, pro nějž je definované omezení FOREIGN KEY, nemusí mít stejný název jako ten odkazovaný sloupec v druhé tabulce, nicméně platí, že oba tyto sloupce musí být stejného datového typu a musí u nich být definováno stejné omezení týkající se použití hodnoty NULL.

K zobrazení struktury nějaké existující tabulky (a tedy i k ověření její existence) můžeme použít příkazy DESCRIBE či SHOW CREATE TABLE. Příkaz DESCRIBE vrací seznam názvů sloupců tabulky spolu s jejich datovými typy a omezeními, zatímco příkaz SHOW CREATE TABLE vrací příkaz, pomocí něhož by bylo možné tabulku znovu vytvořit.

```
DESCRIBE zamestnanec;

SHOW CREATE TABLE zamestnanec;
```

Tip: Zvolte si nějakou konvenci

Postupem času jsem při své práci se serverem MySQL začal používat jednoduchou konvenci: všechna klíčová slova serveru MySQL píši velkými písmeny, zatímco svoje vlastní identifikátory píši malými písmeny. Při zpracovávání klíčových slov a názvů sloupců server MySQL nerozlišuje velká a malá písmena, avšak v případě názvů tabulek již to tak jednoduché není a závisí to na souborovém systému, v němž jsou uloženy soubory vašich tabulek. Optimálním řešením tedy je zvolit si nějakou – prakticky libovolnou – konvenci či standard a ty dále dodržovat.

Až dosud jsme se bavili o atributech sloupců a omezeních tabulek uvedených v našem příkladu, a neřekli jsme si tedy nic o datových typech. Možná se vám bude zdát, že následující část této kapitoly je poněkud suchá a teoretická. Je však poměrně důležitá, neboť se věnuje zásadnímu tématu. Jak asi víte, každý datový typ vyžaduje odlišné množství prostoru na

pevném disku, přičemž při navrhování databází má většinou každý tendenci použít jen ten nezbytně nutný typ. Množství ztraceného diskového prostoru vyplývající z použití takového datového typu, který by byl ve srovnání s ukládanými daty zbytečně veliký, se vám může zdát na první pohled zanedbatelné, protože zpočátku budete mít v tabulce pouze pár záznamů. Jenomže jakmile začnete do takové tabulky přidávat další a další záznamy, bude toto množství neustále narůstat.

Datové typy a jejich požadavky na úložiště

Server MySQL podporuje mnoho datových typů, z nichž většinu probereme v následujících odstavcích. Pojmem **datový typ** se označuje klasifikace dat vycházející z hodnot, které je možné použít, ze sady operací, které je možné s takovými daty provádět, a z jejich požadavků na úložiště. Tak například hodnoty datového typu `INTEGER` smí být tvořeny pouze celými čísly, například 0, 42 či 1337. Tento datový typ se tedy liší od typu `DECIMAL`, tvořeného čísly s plovoucí desetinnou čárkou, například 1,61, 3,14 či 100,0. Hodnoty typu `INTEGER` či `DECIMAL` můžeme sčítat, odčítat, násobit nebo dělit, což jsou všechno operace, které nemůžeme provádět s textovými datovými typy, jimiž jsou například `CHAR` či `TEXT`.

Číselné datové typy

Server MySQL podporuje datové typy `INTEGER` (který bývá též zkracován jako `INT`), `TINYINT`, `SMALLINT`, `MEDIUMINT` a `BIGINT`. Všechny tyto typy jsou určeny pouze pro ukládání celých čísel a vzájemně se liší počtem bajtů potřebným k uložení jedné hodnoty. Tento počet bajtů na druhou stranu zase určuje ten rozsah celých čísel, s nímž je možné v každém datovém typu pracovat. Tak například `TINYINT` využívá 1 bajt, díky čemuž je jeho rozsah omezen hodnotami -128 a 127 – to je totiž rozsah čísel, která je možné v binární podobě vyjádřit pomocí 8 bitů. Naopak datový typ `INTEGER` pracuje se 4 bajty, a proto je jím nabízený rozsah možných hodnot podstatně větší: -2 147 483 648 až 2 147 483 647.

V případě celočíselných datových typů můžeme použít i atribut `UNSIGNED`. Takto modifikovaný datový typ využívá zcela stejný prostor, avšak používání záporných hodnot není povoleno, a proto je nejvyšší možná kladná hodnota vyšší. Tak například rozsah datového typu `TINYINT UNSIGNED` je 0 až 255. Přitom platí, že jak datový typ `TINYINT`, tak i `TINYINT UNSIGNED` umožňují použití 256 různých hodnot, avšak počátky těchto rozsahů se liší: buď je to -128, anebo je to 0.

V následující tabulce vidíte požadavky na úložiště a rozsahy možných hodnot jednotlivých celočíselných datových typů podporovaných serverem MySQL. Uvedeny jsou přitom obě varianty, tj. jak se znaménkem (výchozí), tak i bez znaménka (`UNSIGNED`):

Datový typ	Využívá bajtů	Minimum se znaménkem	Maximum se znaménkem	Minimum bez znaménka	Maximum bez znaménka
`TINYINT`	1	-128	127	0	255
`SMALLINT`	2	-32 768	32 767	0	65 535

Datový typ	Využívá bajtů	Minimum se znaménkem	Maximum se znaménkem	Minimum bez znaménka	Maximum bez znaménka
MEDIUMINT	3	-8 388 608	8 388 607	0	6 777 215
INTEGER	4	-2 147 483 648	2 147 483 647	0	4 294 967 295
BIGINT	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	0	18 446 744 073 709 551 615

Datové typy DECIMAL, FLOAT a DOUBLE umožňují práci s čísly s plovoucí desetinnou čárkou. V jejich případě je tedy nutné zadat jak požadovaný rozsah (tj. celkový počet číslic), tak i přesnost (tj. počet číslic následujících za desetinnou čárkou). Například datový typ DECIMAL (5,2) nabízí rozsah hodnot -999,99 až 999,99 – tj. celkem 5 číslic, z nichž 2 musí následovat za desetinnou čárkou. Přitom je nutné zdůraznit, že při vlastním zápisu desetinných hodnot musíte v MySQL používat jako oddělovač tečku, nikoliv čárku, jak odpovídá zvyklostem v ČR. I v případě těchto datových typů je možné použít atribut UNSIGNED, který však znemožní pouze používání záporných hodnot. To proto, že horní limit je definován námi zadaným rozsahem a požadovanou přesností.

Specifikem datového typu DECIMAL je, že je to vlastně typ s pevnou desetinnou čárkou – jinými slovy řečeno, při výpočtech s hodnotami tohoto typu je neustále udržována zadaná přesnost. Toto je užitečné zejména ve chvíli, kdy pracujete s hodnotami představujícími například nějaké peněžní částky. Maximální rozsah tohoto datového typu je 65 a maximální přesnost je 30, tj. můžeme pracovat nejvýše s číslem majícím celkem 65 číslic, z nichž 30 může být za desetinnou čárkou. Na druhou stranu datové typy FLOAT a DOUBLE představují typy s plovoucí desetinnou čárkou. Výpočty s těmito datovými typy jsou pouze přibližné, neboť při každém z nich může dojít k nějakému zaokrouhlení způsobenému tím, jak je hodnota interně v počítači reprezentována. Rozdílem mezi datovými typy FLOAT a DOUBLE pak je prostor, který potřebují k uložení jednotlivých hodnot, přičemž ten zase ovlivňuje jejich rozsah. Datový typ FLOAT vyžaduje pro uložení každé hodnoty 4 bajty a považuje se za přesný až do 7 desetinných míst. Naopak datový typ DOUBLE vyžaduje pro uložení každé hodnoty 8 bajtů a považuje se za přesný až do 15 desetinných míst.

Posledním číselným datovým typem, jímž se budeme zabývat, je BIT. Ten je určený pro uložení posloupnosti bitů, a je tedy vhodný například pro práci s návěštími či bitovými maskami. Datový typ BIT může pracovat s 1 až 64 bity. Do sloupce definovaného jako BIT (1) tedy můžete uložit pouze hodnotu 0 nebo 1; do sloupce definovaného jako BIT (2) můžete uložit hodnoty 00, 01, 10 či 11; do sloupce definovaného jako BIT (3) můžete uložit binární hodnoty 000, 001, 010, 011, 100, 101, 110 či 111; atd. Server MySQL využívá k označení hodnoty jako řetězce binárních číslic zápis ve formátu b'hodnota', tedy například b'101010'.

Řetězcové datové typy

Server MySQL věnuje práci s řetězcovými či textovými daty několik typů: `CHAR`, `VARCHAR`, `BINARY`, `VARBINARY`, `TEXT`, `TINYTEXT`, `MEDIUMTEXT`, `LONGTEXT`, `BLOB`, `TINYBLOB`, `MEDIUMBLOB` a `LONGBLOB`. Velikostí omezené typy `TINYTEXT` a `MEDIUMTEXT` se chovají úplně stejně jako typ `TEXT` s tím, že každý z nich je omezen odlišným maximálním množstvím textu, který do něj lze uložit. Totéž platí i pro datový typ `BLOB` a jeho velikostí omezené odvozeniny `TINYBLOB`, `MEDIUMBLOB` a `LONGBLOB`.

Při práci s datovými typy `CHAR` a `VARCHAR` musíme vždy zadat délku. Například do sloupce definovaného jako `CHAR (255)` lze ukládat textové řetězce o délce 255 znaků, zatímco do sloupce definovaného jako `VARCHAR (255)` lze ukládat textové řetězce o délce až 255 znaků. Všimněte si, že jednou jsem řekl „255 znaků", zatímco podruhé „až 255 znaků". To proto, že datový typ `CHAR` je navržen pro ukládání řetězců pevné délky, tj. textových hodnot, které budou mít ve všech záznamech tabulky stejnou délku. To znamená, že prostor pro uložení každé z takových hodnot je stále stejný. Naopak datový typ `VARCHAR` umožňuje ukládání textových řetězců s proměnnou délkou, což mimo jiné znamená, že v každém záznamu tabulky může být hodnota jiné délky. Prostor potřebný pro uložení každé takové hodnoty je pak dán délkou řetězce.

Rozdíl mezi datovými typy `CHAR` a `VARCHAR` si můžeme názorně ukázat na známém řetězci „Ahoj světe". Tento řetězec má délku 10 znaků, a proto pokud jej uložíme do sloupce definovaného jako `VARCHAR (255)`, bude pro jeho uložení nezbytných 10 bajtů (plus jeden či dva bajty navíc, které server MySQL potřebuje z důvodu vlastní režie). Použijeme-li ovšem sloupec `CHAR (255)`, bude pro jeho uložení použita plná délka 255 znaků, tj. MySQL doplní námi zadaný řetězec celkem 245 mezerami. Při načtení takto uložené hodnoty budou doplněné mezery opět odstraněny a server MySQL vrátí původní řetězec „Ahoj světe". Jak jsme si však řekli výše, každý řetězec uložený do sloupce `CHAR (255)` bude zabírat 255 bajtů.

Poznámka: Maximální délky

Datové typy **CHAR** a **VARCHAR** mají rozdílné maximální délky. Zatímco do sloupce typu **CHAR** můžete uložit až 255 znaků, do sloupce typu **VARCHAR** je to až 65 535 znaků. Je však pravděpodobné, že sloupec definovaný jako **VARCHAR (65535)** stejně nikdy nepoužijete. Server MySQL totiž omezuje maximální velikost celého řádku tabulky na 65 535 bajtů. A protože naprostá většina sloupců se do tohoto limitu započítává (výjimku představují datové typy **TEXT** a **BLOB** a jejich odvozeniny, z nichž se do velikosti řádku započítává pouze několik bajtů), nezbyl by vám při použití takového sloupce žádný prostor na ostatní potřebné sloupce. Podrobné informace týkající se limitů velikosti řádků a sloupců najdete v online dokumentaci[3].

Datové typy `BINARY` a `VARBINARY` se chovají podobně jako `CHAR` a `VARCHAR` s tím rozdílem, že jsou určeny pro ukládání binárních řetězců. Přitom server MySQL považuje binární řetězce za řady bajtů, nikoliv znaky, a proto při práci s nimi neuvažuje collation ani znakovou sadu.

3 *https://dev.mysql.com/doc/refman/5.7/en/column-count-limit.html*

Navíc není používána ani žádná speciální sémantika: veškeré operace řazení či porovnávání jsou prováděny na základě ordinálních hodnot jednotlivých bajtů. Pro doplňování datového typu `BINARY` využívá server MySQL bajt `NULL` (`0x00`).

Datové typy `TEXT` a `BLOB` jsou datovými typy proměnné délky určenými pro ukládání většího množství textu. Žádný z nich uložený text nedoplňuje mezerami či jakýmikoliv jinými hodnotami, díky čemuž se jedná o datové typy optimální pro ukládání textu přesně v té podobě, v jaké byl zadán. Hodnoty typu `TEXT` jsou zpracovávány jako řetězce znaků, zatímco hodnoty typu `BLOB` jsou zpracovávány jako binární řetězce. A jak jsme si řekli v poznámce výše, sloupce typu `TEXT` či `BLOB` (a jejich velikostně omezené varianty) jsou v podstatě vyloučeny z výpočtu velikosti řádku. Pokud tedy potřebujete uložit do jednoho řádku tabulky více než 65 535 bajtů, zamyslete se nad použitím jednoho z těchto typů.

Nebinárním řetězcovým typům `CHAR`, `VARCHAR` a `TEXT` může být přiřazen atribut `CHARACTER SET`, určující kódování znaků. **Znaková sada** (angl.. character set) říká, jakým způsobem jsou zadané bity a bajty interpretovány při jejich převodu do čitelných znaků. Mezi často používané znakové sady patří ASCII (`ascii`), ISO 8859-1 (`latin1`), ISO 8859-2 (`latin2`) či UTF-8 (`utf8`). Nezadáte-li při definici sloupce žádnou znakovou sadu, bude použita výchozí, jíž je `latin1`. Seznam podporovaných znakových sad si můžete také zobrazit příkazem `SHOW CHARACTER SET;`.

```
CREATE TABLE charset_priklad (
    id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    ascii_retezec VARCHAR(255) CHARACTER SET ascii NOT NULL,
    latin2_retezec VARCHAR(255) CHARACTER SET latin2 NOT NULL,
    utf8_retezec VARCHAR(255) CHARACTER SET utf8 NOT NULL,
    PRIMRAY KEY (id)
);
```

Datové typy `ENUM` a `SET` říkají, že kterákoliv ze zadaných hodnot musí odpovídat jedné z hodnot definovaného seznamu. Tak například sloupec `ENUM('Alfa','Beta','Gamma')` smí obsahovat pouze jeden z řetězců vyjmenovaných v definici, tj. „Alfa“, „Beta“ či „Gamma“. Naopak do sloupce typu `SET` je možné uložit jednu či více čárkami oddělených hodnot z definovaného seznamu. Jinými slovy řečeno, ve sloupci `SET('Alfa','Beta','Gamma')` smí být uloženy hodnoty „Alfa“, „Beta,Gamma“, „Alfa,Beta,Gamma“ apod.

Upozornění: Datový typ ENUM používejte s rozvahou

Pokud to vyhovuje potřebám mé aplikace, nemám nic proti použití datového typu **`ENUM`**. Mějte však na paměti, že tento zdánlivě nevinný datový typ je provázen určitou kontroverzí. Více informací o tomto tématu najdete v příspěvku s názvem „8 Reasons Why MySQL's **`ENUM`** Data Type Is Evil“[4], který na svém blogu zveřejnil Chris Komlenic.

4 *http://komlenic.com/244*

V následující tabulce vidíte přehled řetězcových typů, podporovaných serverem MySQL spolu s informacemi o jejich požadavcích na úložiště a maximální možné délce:

Datový typ	Využívá bajtů	Maximální délka
`CHAR(m)`	`m` × znak maximální velikosti v použité znakové sadě	255 znaků
`VARCHAR(m)`	až 2 bajty + `m` × znak maximální velikosti v použité znakové sadě	65 535 znaků
`TEXT`	velikost řetězce v bajtech + 2	65 535 znaků
`TINYTEXT`	velikost řetězce v bajtech + 1	255 znaků
`MEDIUMTEXT`	velikost řetězce v bajtech + 3	16 777 215 znaků
`LONGTEXT`	velikost řetězce v bajtech + 4	4 294 967 295 znaků
`BINARY(m)`	`m`	255 bajtů
`VARBINARY(m)`	až 2 bajty + `m`	65 535 bajtů
`BLOB`	velikost řetězce v bajtech + 2	65 535 bajtů
`TINYBLOB`	velikost řetězce v bajtech + 1	255 bajtů
`MEDIUMBLOB`	velikost řetězce v bajtech + 3	16 777 215 bajtů
`LONGBLOB`	velikost řetězce v bajtech + 4	4 294 967 295 znaků
`ENUM`	až 2 bajty	65 535 hodnot
`SET`	až 8 bajtů	64 členů

Datové typy pro práci s datem a časem

Datové typy `DATETIME`, `TIMESTAMP`, `DATE`, `TIME` a `YEAR` jsou určeny pro práci s datem a časem. Jak datový typ `DATETIME`, tak i datový typ `TIMESTAMP` umožňují ukládání hodnot popisujících datum a čas ve formátu `'RRRR-MM-DD HH:mm:ss'` (kde `RRRR` představuje rok zadaný čtyřmi číslicemi, `MM` představuje měsíc zadaný dvěma číslicemi, `DD` představuje den zadaný dvěma číslicemi, `HH` představuje hodinu zadanou dvěma číslicemi, `mm` představuje minuty zadané dvěma číslicemi a konečně `ss` představuje sekundy zadané dvěma číslicemi). Tak například `'2016-03-15 13:15:00'` odpovídá datu 15. března 2016 a času 13:15. Datové typy `DATE`, `TIME` a `YEAR` pak umožňují práci s odpovídajícími částmi celého data. Kromě údaje `'HH:mm:ss'` vyjadřujícího hodinu dne – například `'13:15:00'` – mohou hodnoty typu `TIME` obsahovat i údaje ve formátu `'HHH:mm:ss'` vyjadřující čas uplynulý od nějaké události. Typickým příkladem může být údaj `'293:23:10'`, znamenající 293 hodin 23 minut a 10 sekund.

Kdykoliv je do tabulky vkládán nový řádek či je stávající řádek aktualizován, je server MySQL schopen automaticky inicializovat a aktualizovat hodnoty typu `DATETIME` a `TIMESTAMP` aktuálním datem a časem. Jinými slovy řečeno, vložíte-li do nějaké tabulky nový záznam příkazem `INSERT` a nezadáte-li žádný údaj pro sloupec typu `TIMESTAMP`, server MySQL automaticky použije aktuální datum a čas. Server MySQL také automaticky aktualizuje hodnotu ve sloupci typu `TIMESTAMP` ve chvíli, kdy je aktualizována kterákoliv z hodnot

v příslušném řádku. Toto chování je zcela automatické pro sloupce typu `TIMESTAMP`. Pokud bychom však chtěli, aby server MySQL takto pracoval i se sloupci typu `DATETIME`, museli bychom při definici příslušných sloupců použít atributy `DEFAULT CURRENT_TIMESTAMP` a `ON UPDATE CURRENT_TIMESTAMP`.

Upozornění: Odchylky v datových typech pro práci s datem a časem

Datové typy pro práci s datem a časem se v jednotlivých verzích serveru MySQL chovají poněkud odlišně. Toto platí zejména pro typy `TIMESTAMP` a `YEAR`. Chcete-li se dozvědět více, podívejte se do online dokumentace na témata týkající se inicializace[5] časového razítka a práce s roky zadanými pomocí pouze dvou číslic[6].

V následující tabulce vidíte přehled všech datových typů serveru MySQL určených pro práci s datem a časem. U každého z nich najdete i informace o potřebném úložišti a minimální, resp. maximální možné hodnotě:

Datový typ	Využívá bajtů	Minimální hodnota	Maximální hodnota
`DATETIME`	8	1000-01-01 00:00:00	9999-12-31 23:59:59
`TIMESTAMP`	4	1970-01-01 00:00:01 UTC	2038-01-19 03:14:07 UTC
`DATE`	3	1000-01-01	9999-12-31
`TIME`	3	-838:59:59	838:59:59
`YEAR`	1	1901	2155

Datové typy pro práci s informacemi o prostoru

Konsorcium Open Geospatial Consortium publikovalo celou řadu standardů zaměřených na reprezentaci geografických informací v různých formátech. Server MySQL vám nabízí několik datových typů vycházejících z práce tohoto konsorcia a určených pro ukládání takových dat: `POINT`, `LINESTRING`, `POLYGON`, `GEOMETRY`, `MULTIPOINT`, `MULTISTRING`, `MULTIPOLYGON` a `GEOMETRYCOLLECTION`.

Datové typy `POINT`, `LINESTRING` a `POLYGON` podporují práci s geometrickými hodnotami odpovídajícími jejich názvům. Do datového typu `POINT` lze tedy uložit souřadnice jediného bodu, například `Point(10, 20)`. Naopak typ `LINESTRING` je určen pro ukládání dvojic bodů definujících úsečku, například `LineString(Point(10, 20), Point(30, 20))`. Pokud se týká typu `GEOMETRY`, ten lze označit za jakýsi „nadřazený" typ, neboť do něj lze ukládat jak hodnoty typu `POINT`, tak i hodnoty typu `LINESTRING`. Zbývají nám všechny datové typy začínající `MULTI` a typ `GEOMETRYCOLLECTION`. Na rozdíl od odpovídajících typů určených pro práci s informacemi o jediném bodu, jediné úsečce apod. lze do těchto ukládat informace o více členech téhož typu. Například do sloupce typu `MULTIPOINT` lze uložit informace o několika bodech: `MultiPoint(Point(10, 20), Point(20, 20), Point(30, 20))`.

5 *http://dev.mysql.com/doc/refman/5.7/en/timestamp-initialization.html*

6 *http://dev.mysql.com/doc/refman/5.7/en/two-digit-years.html*

Lze však říci, že práce s informacemi o prostoru představuje samostatnou disciplínu, a proto se jí zde nebudeme dále věnovat. Následující zdroje jsou dobrým výchozím bodem pro ty z vás, kteří se chtějí dozvědět něco více o možnostech, které vám pro práci s informacemi o prostoru nabízí server MySQL:

- Dokumentace MySQL: rozšíření pro data popisující prostor[7]
- GIS fórum MySQL[8]
- Open Geospatial Consortium[9]

Databázová úložiště

Pojmem **databázové úložiště** se označuje základní logika a veškerý programový kód odpovědný za správu tabulek a dat. Databázové úložiště nám v podstatě umožňuje nezabývat se specifickými podrobnostmi týkajícími se toho, jak data mohou být uspořádána na pevném disku či v paměti, a nadále si je představovat pouze pomocí tabulek a jejich záznamů. Platí však, že rozdílná databázová úložiště mají rozdílné možnosti a rozdílná omezení. Server MySQL implementuje architekturu zásuvných modulů, díky níž si v souladu se svými aktuálními potřebami můžeme vybírat z několika podporovaných databázových úložišť. Přitom platí, že dokonce i v jedné databázi mohou být jednotlivé tabulky spravovány rozdílnými úložišti.

Chcete-li si prohlédnout seznam databázových úložišť podporovaných vaší instalací serveru MySQL, spusťte příkaz `SHOW ENGINES`:

```
SHOW ENGINES;
```

InnoDB je výchozím databázovým úložištěm serveru MySQL již od verze 5.5. To znamená, že spustíte-li příkaz `CREATE TABLE` neobsahující klauzuli `ENGINE`, bude výsledkem tabulka spravovaná úložištěm InnoDB. Úložiště InnoDB je kompatibilní s ACID, což znamená, že každá transakce – jeden či více příkazů odeslaných na server, které musí být zpracovány jako jeden celek – musí dodržovat následující čtyři omezení:

- **Nedělitelnost** (angl. atomicity) – každá transakce je buď provedena a dokončena jako celek, anebo není provedena vůbec. Pokud část transakce skončí neúspěšně (řekněme, že například v polovině dávky dojde k chybě), data musí být automaticky vrácena do toho stavu, v němž se nacházela před spuštěním transakce.
- **Konzistence** (angl. consistency) – všechny změny provedené transakcí musí data ponechat v platném stavu odpovídajícím veškerým omezením a dalším pravidlům. To mimo jiné znamená, že nesmí dojít k žádnému porušení omezení daného cizím klíčem či že se v žádném sloupci nesmí nacházet nějaké nepovolené hodnoty (například hodnota `NULL` ve sloupci definovaném pomocí atributu `NOT NULL`) apod.

7 *http://dev.mysql.com/doc/refman/5.7/en/spatial-extensions.html*

8 *http://forums.mysql.com/list.php?23*

9 *http://www.opengeospatial.org/*

- **Izolace** (angl. isolation) – všechny transakce musí probíhat zcela nezávisle na sobě. Jinými slovy řečeno, dvě transakce probíhající ve shodném čase se nesmí žádným způsobem vzájemně ovlivňovat.
- **Trvalost** (angl. durability) – jakmile jsou nějaké změny v rámci transakce provedeny a dokončeny, jsou konečné a nelze je již vrátit zpět.

Úložiště InnoDB také využívá uzamykání na úrovni záznamů. Kdykoliv totiž nějaký proces tohoto úložiště zapisuje do tabulky záznam, musí nějakým způsobem zajistit, že jiný proces se v tomtéž čase nepokusí o zápis do téhož záznamu. Ve snaze o zabránění vzniku takových situací zapisující proces vytvoří zámek, zapíše data a teprve poté zámek zase uvolní. Druhý proces pokoušející se o přístup k témuž záznamu pak musí čekat až do doby uvolnění zámku. Teprve pak bude moci vytvořit vlastní zámek. Přitom platí, že několik procesů může současně přidávat data do téže tabulky či je v ní aktualizovat, pouze nesmí pracovat se shodným záznamem.

Díky své kompatibilitě s ACID a strategií uzamykání na úrovni záznamů je InnoDB vhodnou volbou vždy, když potřebujete spolehlivé a obecně použitelné databázové úložiště.

Před verzí MySQL 5.5 bylo výchozím úložištěm úložiště MyISAM. To vychází z původního úložiště ISAM databáze UNIREG, které bylo poněkud upraveno pro potřeby MySQL. Z tohoto důvodu se toto úložiště zaměřuje především na vysokou rychlost a schopnost pracovat i s menšími prostředky. Úložiště MyISAM vám ovšem nabízí vysokou rychlost za cenu toho, že není kompatibilní s ACID a neumožňuje práci s cizími klíči. Jelikož používání úložiště MyISAM není provázeno režií vyplývající z dodržování standardu ACID, je toto úložiště výhodnou volbou především v případě, že potřebujete nějaká data nabízet pouze ke čtení, anebo v případě, že potřebujete provozovat MySQL na méně výkonném hardwaru.

Úložiště MyISAM využívá uzamykání na úrovni tabulek. To znamená, že zámek je vytvářen na úrovni celé tabulky. Díky tomu žádné dva procesy nejsou schopny najednou přidávat či aktualizovat data ve stejné tabulce. Z toho vyplývá, že úložiště MyISAM vám nabízí nižší úroveň souběžnosti než úložiště InnoDB.

Kromě úložišť InnoDB a MyISAM podporuje server MySQL ještě další:

- **Archive** – úložiště Archive je výhodné tehdy, pokud potřebujete uložit velké množství dat, k nimž budete přistupovat pouze velmi zřídka. Nové záznamy jsou komprimovány a připojovány ke stávajícím datům tabulky. Jednotlivé záznamy jsou pak opětovně dekomprimovány až v případě potřeby jejich načtení. Díky komprimaci zaujímají data podstatně menší prostor na disku, nicméně díky ní také server MySQL nemůže využívat indexy pro efektivní vyhledávání dat. Důsledkem je to, že server MySQL musí při vyhledávání odpovídajících záznamů sekvenčně procházet celé tabulky. A strategie připojování zkomprimovaných záznamů také znamená, že při použití tohoto úložiště nejsou podporovány příkazy typu `DELETE`, `UPDATE` a `REPLACE`.

- **Blackhole** – úložiště Blackhole se chová podobně jako tradiční unixové zařízení `/dev/null`. Toto úložiště tedy přijímá veškerá data, která do něj odešleme, a vzápětí je zahazuje. Vypůjčíme-li si slogan od jednoho známého výrobce pastí na hmyz, „data se sice dostanou dovnitř, ale nikdy ne ven". Toto úložiště je výhodné zejména při vývoji a testování a pak také v případech, kdy v replikovaném prostředí potřebujete připravit systém opakovače.
- **CSV** – úložiště CSV ukládá data do textových souborů, přičemž platí, že data z jednotlivých sloupců jsou vzájemně oddělena čárkami. Takové soubory lze pak snadno zpracovávat v tabulkových kalkulátorech, mezi něž patří např. Microsoft Excel či OpenOffice Calc, a proto je toto úložiště výhodné zejména tehdy, potřebujete-li přesouvat data tabulky do či z tabulkového kalkulátoru anebo je exportovat do nějakého automatizovaného procesu. Nicméně z důvodu zajištění integrity CSV souborů platí pro toto úložiště celá řada omezení: nepodporuje práci s primárními a cizími klíči, různými omezeními či se sloupci umožňujícími ukládání hodnoty `NULL`.
- **Federated** – úložiště Federated je výhodné tehdy, potřebujete-li pracovat s daty nějaké vzdálené tabulky serveru MySQL tak, jako kdyby to byla data lokální. V případě tabulky spravované úložištěm Federated nejsou žádná data ukládána lokálně. Namísto toho jsou všechny záznamy načítány ze vzdálené databáze a do ní jsou také zapisovány. Mějte však na paměti, že práce se vzdálenými daty je pomalejší než práce s lokálními daty a není podporováno ani používání indexů, neboť samotný přístup k datům je řízen vzdáleným serverem.
- **Merge (MRG_MyISAM)** – pro velikost tabulek spravovaných úložištěm MyISAM platí veškerá omezení vyplývající ze souborového systému použitého daným operačním systémem. Úložiště Merge nám umožňuje obejít tato omezení tím, že pomocí něho můžeme sloučit několik tabulek spravovaných úložištěm MyISAM dohromady a pracovat pak se záznamy tak, jako kdyby byly uloženy v jedné rozsáhlé tabulce. Platí však, že každá tabulka, která je součástí této sloučené tabulky, musí mít stejné pořadí sloupců, jejichž definice musí být také shodné.
- **Memory** – úložiště Memory ukládá data do operační paměti, což přináší nižší zpoždění a rychlejší přístup k datům než v případě použití úložiště založeného na využití pevného disku. Nicméně platí, že do tabulky spravované tímto úložištěm byste měli ukládat pouze nějaká podpůrná či nedůležitá data, neboť v okamžiku restartu serveru MySQL budou tato data ztracena. Z důvodu velké rychlosti přístupu je toto úložiště výhodné zejména pro ukládání takových věcí jako například mezipaměti či dat relací.
- **NDB** – jedná se o úložiště navržené pro podporu clusteringu databází. Podobně jako InnoDB i NDB je kompatibilní se standardem ACID.

Více podrobností o každém z uvedených úložišť najdete v online dokumentaci[10]. Nicméně abychom vám pomohli pochopit, jaké kompromisy dělají jednotlivá úložiště proto, aby

10 *http://dev.mysql.com/doc/refman/5.7/en/storage-engines.html*

například zajistila vyšší výkon či naopak podporu určitých specifických funkcí, připravili jsme pro vás tabulku, v níž jsou zdůrazněny některé klíčové rozdíly mezi výše popsanými úložišti. V případě funkcí závisejících na úložišti spravujícím základní tabulku je použit otazník (?).

Funkce	InnoDB	MyISAM	Archive	Black-hole	CSV	Federa-ted	Merge	Memory	NDB
Limit úložiště	64 TB	256 TB	Bez limitu	Neexis-tuje	Souborový systém	Neexis-tuje	Bez limitu	Velikost RAM	384 EB
Kompati--bilita s ACID	Ano	Ne	Ne	Ne	Ne	?	Ne	Ne	Ano
Vynucení cizích klíčů	Ano	Ne	Ne	Ne	Ne	?	Ne	Ne	Ano
Indexy	Ano	Ano	Ne	Neexis-tuje	Ne	Ne	Ano	Ano	Ano
Současné vkládání záznamů	Ano	Ano	Ano	Ano	Ano	?	Ano	Ne	Ano
Vynucení omezení UNIQUE	Ano	Ano	Ne	Ne	Ne	?	Ne	Ano	Ano
Fulltexto-vé vyhledá-vání	Ano	Ano	Ne	Neexis-tuje	Ne	?	Ne	Ne	Ne
Úroveň zámků	Záznam	Tabulka	Tabulka	Tabulka	Tabulka	?	Tabulka	Tabulka	Záznam
Transakce	Ano	Ne	Ne	Ne	Ne	?	Ne	Ne	Ano
Datové typy pro práci s prostorem	Ano	Ano	Ano	Ano	Ne	?	Ne	Ne	Ano
Šifrování	Ano	Ano	Ano	Neexis-tuje	Ano	Ano	Ano	Ano	Ano
Replikace	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ano
Clustering databází	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ano

Přidávání dat

Pro přidávání nových záznamů do tabulky slouží příkaz `INSERT`. Lze říci, že složitost tohoto příkazu je podstatně menší než složitost příkazu `CREATE TABLE` – což je něco, co po čtení několika předcházejících stran zcela určitě slyšíte rádi.

Následujícím příkazem INSERT přidejte nový záznam do tabulky zamestnanec. Jak vidíte, součástí příkazu je cílová tabulka, seznam sloupců, do nichž chceme data vložit, a samotné vkládané hodnoty.

```
INSERT INTO zamestnanec
    (jmeno, prijmeni, email, datum_nastupu)
VALUES
    ('Jan','Novak','jnovak@cpress.cz','2016-02-22')
```

Příkazu SELECT se sice budeme podrobně věnovat až v následující kapitole, nicméně v tuto chvíli by nám mohlo stačit, že pro načtení obsahu tabulky zamestnanec a pro kontrolu toho, zda jsme data vložili správně, lze použít následující příkaz:

```
SELECT * FROM zamestnanec;
```

Odpověď, zobrazená v klientu příkazového řádku, by měla vypadat zhruba takto:

```
+-------------+---------+------+-----------------+--------------+---------+
| osobni_cislo| prijmeni| jmeno| email           | datum_nastupu| poznamka|
+-------------+---------+------+-----------------+--------------+---------+
|           1 | Novak   | Jan  | jnovak@cpress.cz| 2016-02-22   | NULL    |
+-------------+---------+------+-----------------+--------------+---------+
1 row in set (0.00 sec)
```

Příkazem CREATE TABLE jsme při vytváření tabulky zamestnanec definovali sice 6 sloupců, avšak jak vidíte výše, v příkazu INSERT jsme použili pouze 4 z nich. Obsah sloupce osobní_cislo je totiž vypočítán automaticky v důsledku použití atributu AUTO_INCREMENT, díky němuž server MySQL vloží do tohoto sloupce nejbližší hodnotu následující po dosud maximální hodnotě tohoto sloupce. A sloupec poznamka obsahuje hodnotu NULL, čímž nám říká, že neobsahuje nic. Všimněte si také, že v příkazu INSERT jsme sloupce použili v jiném pořadí, než v jakém byly definovány. Platí totiž, že v příkazu INSERT smíte sloupce uvést v libovolném pořadí za podmínky, že stejným způsobem seřadíte i vkládané hodnoty, aby si odpovídaly datové typy (a aby také nedošlo k případné záměně dat).

Poznámka: Názvy sloupců lze vynechat

Pokud do příkazu INSERT vložíte hodnoty pro všechny sloupce tabulky, je možné (nikoliv však doporučované, přinejmenším tedy z důvodu srozumitelnosti) názvy sloupců zcela vynechat. Platí však, že pokud chcete názvy sloupců vynechat, musíte vkládané hodnoty uvést přesně v tom pořadí, v němž byly sloupce definovány příkazem CREATE TABLE.

```
INSERT INTO zaměstnanec
VALUES (NULL, 'Doskocil','Josef','jdoskocil@cpress.cz','2016-02-22', NULL);
```

Číselné hodnoty, například hodnoty typu INTEGER či DECIMAL, a hodnota NULL se zadávají jako „holé“ literály, neuzavřené do apostrofů. Hodnoty všech ostatních datových typů, mezi něž patří například CHAR či DATE, však musí být při zadávání uzavřeny do apostrofů.

Přitom výchozí konfigurace serveru MySQL umožňuje používání jak apostrofů (‘…‘), tak i uvozovek (“…“), nicméně obvykle je preferováno používání apostrofů, neboť v takovém případě nebudou vaše příkazy závislé na určité konfiguraci.

Možná se nyní ptáte, co se stane, pokud řetězec uzavřený do apostrofů sám obsahuje apostrof. Typickým příkladem by mohlo být anglické příjmení O’Brian. V takovém případě musíme použít nějaký **escape** znak, který vložíme před tento apostrof. Díky tomu se server MySQL dozví, že daný apostrof neoznačuje konec řetězce. Tradiční přístup vychází z použití druhého apostrofu, tedy například `'O''Brian'`. Server MySQL však standardně podporuje pro tyto účely i použití zpětného lomítka. Pak by námi uváděné příjmení bylo zapsáno takto: `'O\'Brian'`. Více informací o řetězcích a práci s nimi najdete v online dokumentaci[11].

Zkuste si napsat několik vlastních příkazů `INSERT`. Do tabulky `zamestnanec` přidejte několik dalších záznamů, přičemž při jejich vkládání vždy vynechte hodnotu sloupce `osobni_cislo` tak, jak jsme to udělali výše. Poté si zkuste data načíst, abyste se přesvědčili, jak funguje automatické přiřazování nejbližších následujících hodnot serverem MySQL.

Nyní zkusme vložit nějaký záznam i do tabulky `adresa`. Připomeňme si, že sloupec `osobni_cislo` této tabulky je definován jako cizí klíč odkazující se na tabulku `zamestnanec`. To znamená, že hodnota sloupce `osobni_cislo` v záznamu adresy zaměstnance Jana Nováka musí být shodná s hodnotou sloupce `osobni_cislo` záznamu téhož zaměstnance v tabulce `zamestnanec`. V našem případě se jedná o hodnotu 1, neboť to byl první záznam, který jsme vložili do tabulky `zamestnanec`:

```
INSERT INTO adresa
    (osobni_cislo, ulice, mesto, kraj, psc)
VALUES
    (1, 'Nova 128/1','Brno','JHM','60200');
```

Používání transakcí

Pojmem **transakce** se označuje sada příkazů vytvářejících ucelenou skupinu. Příkaz `START TRANSACTION` celou transakci zahajuje, přičemž server MySQL považuje každý následující příkaz za součást této transakce. Celá transakce se pak ukončuje buď příkazem `COMMIT`, anebo příkazem `ROLLBACK`. Příkaz `COMMIT` vlastně transakci potvrzuje a teprve po jeho provedení budou změny zapsány do příslušných tabulek. Naopak příkaz `ROLLBACK` transakci ruší a během jeho provádění jsou všechny provedené změny odstraněny a tabulky jsou vráceny do stavu, v němž byly před spuštěním příkazu `START TRANSACTION`.

Zkusme nyní využít transakce k provedení malého experimentu zdůrazňujícího koncepty standardu ACID: nedělitelnost, konzistenci, izolaci a trvalost. Nejprve spusťte následující příkazy. Po jejich dokončení sice v tabulce `zamestnanec` uvidíte nového zaměstnance, nicméně zobrazená změna bude pouze dočasná, neboť transakce nebude zatím dokončena:

11 *http://dev.mysql.com/doc/refman/5.7/en/string-literals.html*

```
START TRANSACTION;

INSERT INTO zamestnanec
    (osobni_cislo, prijmeni, jmeno, email, datum_nastupu)
VALUES
    (42, 'Hurvinek','Frantisek','fhurvinek&cpress.cz','2016-02-22');

SELECT * FROM zamestnanec;
```

Nyní si otevřete druhé okno příkazového řádku a z něj se podruhé pomocí klienta příkazového řádku připojte k serveru MySQL. Poté spusťte stejný příkaz `SELECT`, jaký vidíte i na konci předcházející skupiny. Možná vás překvapí to, že v zobrazeném výsledku neuvidíte nově přidaného zaměstnance. Toto nám jasně dokazuje, že transakce, kterou jsme spustili z prvního okna příkazového řádku, je izolována od jakéhokoliv jiného připojení.

Vraťte se zpět do prvního okna příkazového řádku a celou transakce vraťte zpět příkazem `ROLLBACK`. Následný příkaz `SELECT` vám ukáže, že nově přidaný zaměstnanec v tabulce `zamestnanec` nadále neexistuje. Toto nám jasně dokazuje, že během příkazu `ROLLBACK` byla data vrácena zpět do konzistentního stavu: v tabulce `zamestnanec` se nacházejí ty záznamy, které v ní byly i před zahájením transakce, a současně nebyla porušena žádná omezení.

```
ROLLBACK;

SELECT * FROM zamestnanec;
```

Zkusme nyní zahájit jinou transakci, v níž přidáme nějaké další záznamy. Tentokrát však transakci potvrdíme příkazem `COMMIT`:

```
START TRANSACTION;

INSERT INTO zamestnanec
    (osobni_cislo, prijmeni, jmeno, email, datum_nastupu)
VALUES
    (42, 'Hurvinek','Frantisek','fhurvinek&cpress.cz','2016-02-22');

INSERT INTO adresa
    (osobni_cislo, ulice, mesto, kraj, psc)
VALUES
    (42, 'Spejblova 1A','Brno','JHM','60200');

COMMIT;
```

Pokud nyní z obou oken příkazového řádku spustíte příkaz `SELECT`, uvidíte, že informace o novém zaměstnanci byly zapsány do obou tabulek databáze. Oba příkazy `INSERT` byly přitom provedeny jako nedělitelná jednotka – jak jsme si již řekli, transakce musí zajistit provedení buď všeho, anebo ničeho. V našem případě byla během potvrzení transakce

všechna data úspěšně zapsána, což znamená, že v případě nutnosti tato data aktualizovat či odstranit již budete muset použít jiné příkazy.

Shrnutí

V této kapitole jste se naučili vytvářet nové databázové tabulky pomocí příkazu `CREATE TABLE`. Dozvěděli jste se také, jak se pomocí příkazu `INSERT` do takových tabulek přidávají nové záznamy. Vrátíme-li se ještě k příkazu `CREATE TABLE`, zabývali jsme se i otázkou datových typů podporovaných serverem MySQL, otázkou použití atributu `AUTO_INCREMENT` v případě, že chceme, aby server MySQL automaticky do sloupce primárního klíče vkládal po sobě následující celá čísla, a otázkou vhodnosti použití jednotlivých databázových úložišť pro rozdílné scénáře použití. V případě příkazu `INSERT` jste se navíc seznámili i s používáním apostrofů při práci s řetězci a vyzkoušeli jste si práci s transakcemi.

Ještě než budete pokračovat dále, měli byste věnovat určitý čas podrobnému vyzkoušení problematiky, kterou jsme probírali v této kapitole. Zkuste zjistit, co se stane, pokud se pokusíte do tabulky `adresa` přidat nový záznam, jehož hodnota ve sloupci `osobni_cislo` neexistuje v tabulce `zamestnanec`. A co vám server MySQL řekne, budete-li do tabulky `zamestnanec` vkládat nový záznam, jehož hodnota `osobni_cislo` již v tabulce existuje? Podaří se vám do tabulky `adresa` vložit více záznamů se stejnou hodnotou ve sloupci `osobni_cislo`? Pokud ano, co vám to řekne o vztahu mezi tabulkami `zamestnanec` a `adresa`? Zjistěte si také, jak server MySQL zareaguje, pokud se pokusíte vložit do nějakého sloupce hodnotu překračující omezení použitého datového typu.

Se vzrůstajícím počtem záznamů v tabulce budeme ovšem chtít pracovat s daty nějakým lepším způsobem než tím, který jsme si dosud při práci s příkazem `SELECT` předvedli. Proto se v následující kapitole budeme věnovat tematice načítání pouze těch dat, která nás v daném okamžiku zajímají. Kromě toho si také ukážeme, jak se dají existující záznamy v tabulce aktualizovat a jak se z ní dají odstranit. Na konci kapitoly 3 již budete vědět, jak se na serveru MySQL provádějí čtyři základní operace, očekávané u každého databázového serveru: přidávání záznamů (`INSERT`), načítání záznamů (`SELECT`), aktualizace záznamů (`UPDATE`) a výmaz záznamů (`DELETE`).

Rejstřík

E

F

G

H

CH

I

J

K

L

M

N

O

P

R

Ř

S

Š

T

U

V

W

Y

Z

MySQL Okamžitě

Timothy Boronczyk

Překlad: Milan Daněk
Obálka: Martin Sodomka
Odpovědný redaktor: Martin Herodek
Technický redaktor: Jiří Matoušek

Authorized Czech translation of the English edition of Jump Start MySQL, ISBN 9780992461287

Objednávky knih:
http://knihy.cpress.cz
www.albatrosmedia.cz
eshop@albatrosmedia.cz
bezplatná linka 800 555 513
ISBN tištěné verze 978-80-251-4737-5
ISBN e-knihy 978-80-251-4811-2 (1. zveřejnění, 2016)

Vydalo nakladatelství Computer Press v Brně roku 2016 ve společnosti Albatros Media a. s. se sídlem Na Pankráci 30, Praha 4. Číslo publikace 24 012.

1. vydání

ALBATROS MEDIA a.s.