Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Způsoby návrhu databáze nástrojem MySQL Workbench

Plzeň, 2014 Jana Nosková

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 25. června 2014

Jana Nosková

Poděkování

Ráda bych poděkovala vedoucímu bakalářské práce Ing. Martinu Zímovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Abstrakt

Cílem této práce je nalezení způsobů tvorby fyzického datového modelu pomocí databázového nástroje MySQL Workbench. Některým cestám, vedoucím k vytvoření fyzického datového modelu, předchází tvorba logického datového modelu, která je zde také popsána. Nalezla jsem celkem čtyři různé cesty, jak fyzický datový model vytvořit. Z nich je jedna určena pouze pro pokročilejší uživatele, kteří jsou znalí jazyka SQL. Ostatní způsoby mohou použít i začátečníci. Práce obsahuje i rozdíl mezi jednotlivými cestami.

Abstract

The aim of this thesis is to find ways to create physical data model using a database tool MySQL Workbench. Some of the paths leading to the creation of the physical data model, precedes creation of logical data model, which is also described. I found four different ways to create a physical data model. One of these is intended for advanced users who are familiar with SQL. Other methods can be used even for beginners. The work also includes a description of the difference between the paths.

Obsah

Seznam obrázků

1 Úvod

Téměř každý někdy pocítil potřebu ukládat určité množství různých dat, propojovat je a efektivně získávat zpět v požadovaném čase i podobě. Řešení nabízejí databáze.

Někdy není na škodu umět si takovou databázi vytvořit. V současné době existuje vícero nástrojů, které tento proces usnadňují. Ne všichni ale mají čas se s daným programem nejdříve seznámit a naučit se v něm pracovat. Proto vznikla tato práce, která čtenáře seznámí nejen s databázovým systémem MySQL ale také s nástrojem MySQL Workbench.

Cílem práce je popsat postup pro vytvoření fyzického datového modelu různými cestami. K tomu je nutná alespoň základní znalost MySQL a orientace v programu MySQL Workbench.

Teoretická část této práce se týká úvodu do MySQL. Čtenář se dozví nejen základní terminologii, ale také možnosti, jež tento databázový systém nabízí. Následuje seznámení s nástrojem MySQL Workbench. Popisuji zde, jaké služby program poskytuje a k čemu je určen.

V části praktické jsou rozebrány jednotlivé kroky, které vedou k vytvoření databáze. Nejprve tedy volba části nějakého systému, který je používán v reálném světě. Na něj pak aplikuji různé způsoby vytvoření logického i fyzického datového modelu.

Příloha obsahuje podrobný popis instalace MySQL i MySQL Workbench a jednotlivé kroky, které je třeba vykonat před začátkem práce s nimi. Také zde najdeme EER diagram vybraného systému.

2 Úvod do MySQL

MySQL je jeden z nejpoužívanějších a nejoblíbenějších databázových systémů s otevřeným zdrojovým kódem, tzv. open-source[1]. Je to systém, který je schopen ukládat velké množství různých dat a vracet je zpět v požadované podobě a čase. V současné době je vlastněný společností Oracle.

Jako všechny relační databázové systémy i MySQL využívá pro práci s databázemi strukturovaný dotazovací jazyk SQL (Structured Query Language). Ten dovoluje nejen vytvářet databáze či tabulky a jejich atributy, ale také přidávat a zpracovávat data podle konkrétních kritérií. [1]

2.1 Terminologie

Databáze neboli *schéma* se skládá z jedné či více *tabulek* (entit), které jsou vzájemně propojeny *relacemi* (viz kapitola 2.1.1). Entity popisují objekty v reálném světě, o nichž chceme uchovávat informace.

Tabulky jsou tvořeny jednak *atributy,* neboli názvy sloupců, ale také *záznamy,* jež odpovídají řádkům. Každý sloupec popisuje určitou část dat, kterou má každý záznam v tabulce. Pokud z databáze potřebujeme zjistit jen nějaké údaje, použijeme *dotazy*.

Důležitým pojmem je *primární klíč*. Jedná se o atribut či množinu atributů, s jejichž pomocí se jednoznačně identifikují jednotlivé řádky v tabulce. Jednotlivé hodnoty primárních klíčů jsou jedinečné a žádná nesmí být prázdná. Obvykle se do tabulky zavádí navíc jeden atribut (většinou pojmenovaný ID), který plní pouze funkci primárního klíče. Pomocí tohoto atributu přiřadíme každému záznamu automaticky vygenerovaný klíč, většinou nezáporné celé číslo.

Cizí klíč je atribut či množina atributů v tabulce, kterým odpovídá na základě relace primární klíč jiné tabulky.

[1] Jedná se o vlastní zdrojový kód, neboli programovací kód tvořící MySQL, je volně k dispozici všem – lidé mohou přispívat k MySQL, opravovat chyby, vylepšovat program, navrhovat optimalizace.

Pro rychlejší a snadnější vyhledávání v databázích se používají *indexy*.

2.1.1 Relace

Existují tři typy relací:

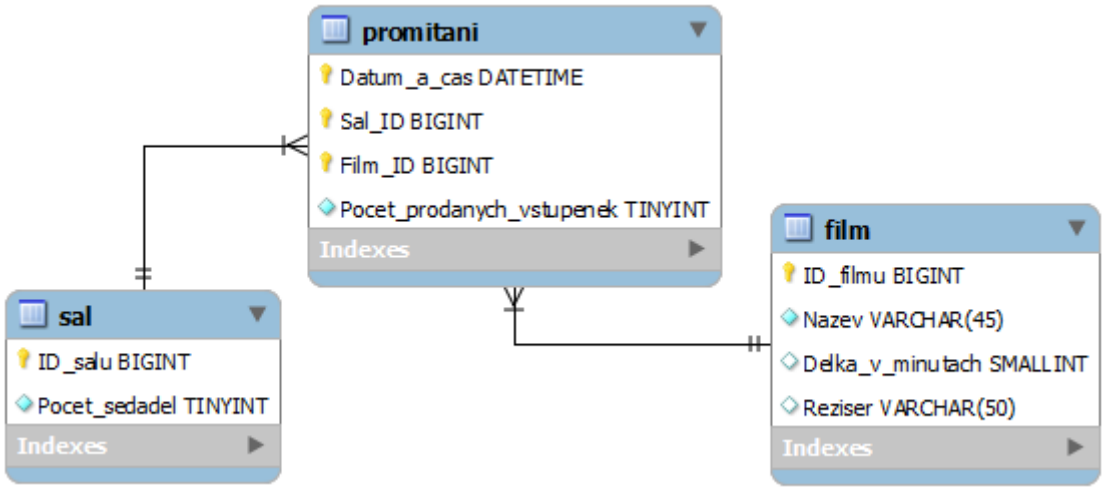
- 1:1
- 1:N (nebo N:1)
- M:N

Relaci 1:1 používáme, pokud záznamu v tabulce A odpovídá maximálně 1 záznam v tabulce B a naopak. Tuto vazbu bychom mohli použít na vztah Prezident – Stát, tedy každý stát má v daném okamžiku pouze jednoho prezidenta. Zároveň však platí, že jeden prezident může v daném okamžiku vládnout pouze jednomu státu.

Mluvíme-li o relaci 1:N, znamená to, že 1 záznamu v tabulce A odpovídá více záznamů v tabulce B. Naopak jednomu záznamu v tabulce B je přiřazen maximálně 1 záznam z tabulky A. Například každá zdravotní pojišťovna má více klientů, ale každý člověk musí být zaregistrován pouze u jedné zdravotní pojišťovny.

Vztah M:N umožňuje každému záznamu v tabulce A přiřadit více záznamů z tabulky B a naopak. Tento vztah je nutné realizovat pomocnou, neboli rozkladovou, tabulkou. Relace musí být rozložena do dvou vztahů 1:N. Relaci M:N lze aplikovat na vazbu Učitel – Student, kdy každý učitel učí více studentů a každý student má několik učitelů (na různé předměty).[3]

U relace 1:1 či 1:N se navíc musí rozhodnout, zda bude tzv. *identifying* či *non-identifying*.



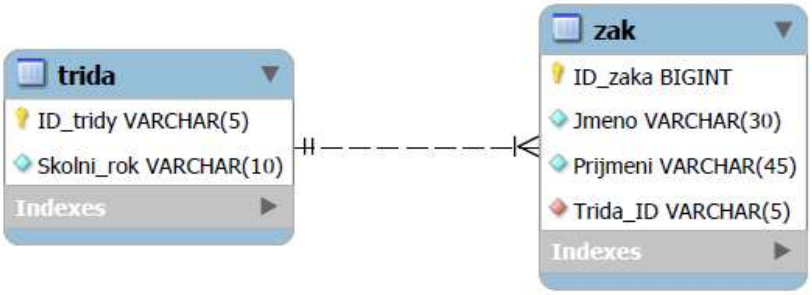
Obrázek 1 - Příklad identifying relace

Pro relaci, která je *identifying*, obecně platí, že primární klíč tabulky A je cizím klíčem tabulky B a zároveň součástí primárního klíče tabulky B.

Konkrétně na Obrázku 1 primární klíč `ID_filmu` v tabulce `film` je zároveň cizím klíčem i součástí primárního klíče v tabulce `promitani`. To samé platí i pro atribut `ID_salu`, který je primárním klíčem v tabulce `sal` a zároveň cizím klíčem i součástí primárního klíče v tabulce `promitani`.

Pro relaci *non-identifying* platí, že primární klíč tabulky A je v tabulce B jen cizím klíčem.

Primární klíč tabulky `trida` je v tabulce `zak` pouze cizím klíčem, jak ukazuje Obrázek 2.



Obrázek 2 - Příklad non-identifying relace

2.2 Datové typy

Při tvorbě databáze se neobejdeme bez znalosti alespoň základních datových typů, jinými slovy typů sloupců. Rozlišujeme tři základní typy: číselné, řetězcové, datum a čas.

Je vhodné volit takový typ sloupce, který poskytuje co nejmenší potřebný rozsah hodnot. Databáze pak bude šetřit místo. Ovšem pozor na volbu typu sloupce s příliš malým rozsahem – může se stát, že data nepůjdou uložit.

2.2.1 Číselné typy

Jsou určeny pro ukládání jakýchkoli číselných dat [1]. Může se jednat o celá čísla i čísla s desetinnou čárkou. Také lze zvolit, zda se číslo uloží bez znaménka (`UNSIGNED`) či se znaménkem (`SIGNED`).

Jestliže je jakékoli pole `UNSIGNED`, znamená to, že neumožňuje uložit záporná čísla. Zároveň však rozšiřuje rozsah hodnot kladných celých čísel.

`TINYINT`
- rozsah hodnot od -128 do +127 (`SIGNED`), bez znaménka (`UNSIGNED`) 0 až 255
- zabírá v paměti 1 byte

`SMALLINT`
- rozsah hodnot od -32 768 do 32 767, bez znaménka 0 až 65 535
- zabírá v paměti 2 byty

`INT` nebo `INTEGER`
- rozsah hodnot od -2 147 483 648 do +2 147 483 647, bez znaménka 0 až 4 294 967 295
- zabírá v paměti 4 byty

`BIGINT`
- rozsah hodnot od -9 223 372 036 854 775 808 do +9 223 372 036 854 775 807 bez znaménka 0 až 18 446 744 073 709 551 615
- zabírá v paměti 8 bytů

`FLOAT`	– číslo s plovoucí desetinnou čárkou jednoduché přesnosti – rozsah hodnot od -3.402823466E+38 do 3.402823466E+38 – zabírá v paměti 4 byty
`DOUBLE`	– číslo s plovoucí desetinnou čárkou dvojnásobné přesnosti – rozsah hodnot od -1.7976931348623157E+308 do 1.7976931348623157E+308 – zabírá v paměti 8 bytů
`DECIMAL (m, d)` nebo `DEC (m, d)`	– číslo s pevnou desetinnou čárkou – rozsah nastavíme parametry "m" (počet číslic včetně des. čárky) a "d" (počet desetinných míst), maximální rozsah je stejný s typem `DOUBLE` – zabírá v paměti 8 bytů

2.2.2 Řetězcové typy

Tento typ se používá pro sloupce, do kterých chceme ukládat libovolná znaková data, například jména, adresy atd. Velikost zabrané paměti se odvíjí od velikosti zadaného řetězce.

`CHAR (m)`	– délka řetězce "m" může být v rozsahu 0-255 – pokud je vložený řetězec kratší než nastavíme, chybějící znaky jsou automaticky doplněny mezerami (má tedy "pevnou" velikost) – `CHAR` (tedy bez "m") je považováno za `CHAR (1)`
`VARCHAR (m)`	– délka řetězce "m" musí být v rozsahu 0-255 – pokud je vložený řetězec kratší než nastavíme, chybějící znaky se nedoplňují

`TEXT` – délka řetězce je maximálně 65 535 znaků
– je prohledáván bez rozlišování velikosti znaků

2.2.3 Typ datum a čas

V případě nutnosti ukládat časové údaje použijeme sloupce typu datum a čas. Jedná se například o datum narození, konkrétní čas atd.

`DATE` – datum ve formátu "rok-měsíc-den" respektive "RRRR-MM-DD" a v rozsahu 1000-01-01 až 9999-12-31

`DATETIME` – datum a čas v rozsahu 1000-01-01 00:00:00 až 9999-12-31 23:59:59 (formát je "RRRR-MM-DD HH:MM:SS")

`TIME` – časový rozsah je od "-838:59:59" do "838:59:59" a formát datového typu "HH:MM:SS"

2.2.4 Ostatní datové typy

`BLOB` – podobný datovému typu `TEXT` – má stejný rozsah i velikost
– slouží pro ukládání binárních dat, např. souborů, obrázků
– je prohledáván s rozlišováním velikosti znaků

`LONGBLOB` – vychází z datového typu `BLOB`
– maximální velikost ukládaných dat je 4 GB

`LONGTEXT` – vychází z řetězcového typu `TEXT`
– maximální velikost ukládaných dat je 4 GB

2.3 Integritní omezení

Pod pojmem integritní omezení si představme pravidla, která pomáhají definovat integritu databáze. Rozeznáváme tři druhy integritních omezení: entitní, doménové a referenční. Většina databázových systémů sama nutí uživatele dodržovat entitní a doménovou integritu. Na referenční integritu však musíme dbát sami.

Integritní omezení dovolí uživateli zadat pouze taková data, která vyhovují předem definovaným kritériím. [2]

Entitní integritní omezení se používá pro specifikaci primárního klíče tabulky. Pomáhá také při označení atributů, které musí obsahovat nějakou hodnotu, nebo při definici jedinečných hodnot.

`NOT NULL` – sloupec s touto vlastností bude muset v každé buňce obsahovat nějakou hodnotu

`PRIMARY KEY`
- označený typ bude sloužit jako primární klíč
- pokud v nástroji MySQL Workbench definujeme některému z atributů tuto vlastnost, automaticky je mu přiřazena i vlastnost `NOT NULL`, protože primární klíč musí vždy obsahovat nějakou hodnotu

`UNIQUE` – v daném sloupci nesmějí být v buňkách stejné hodnoty, tedy co kus to unikát

Doménová integrita umožňuje definovat omezení na úrovni hodnot sloupců. Představme si například omezení rozsahu hodnot.

`CHECK` – kontroluje, zda hodnota atributu leží v zadané množině hodnot

Referenční integritní omezení je mezitabulkovou záležitostí. Definuje vztah mezi dvěma tabulkami s jedním cizím klíčem.[4]

`FOREIGN KEY` – umožní deklarovat cizí klíče

2.4 Jiné vlastnosti

`AUTO_INCREMENT`
- tato vlastnost se používá pouze u MySQL
- systém si sám ve sloupci generuje unikátní (jedinečné) číselné hodnoty
- lze použít pouze na celočíselný datový typ
- za deklarací nové tabulky můžeme ještě navíc určit výchozí hodnotu, např.: `AUTO_INCREMENT=50;`

2.5 Základní příkazy

Jak již bylo zmíněno, systém MySQL využívá pro definici dat a manipulaci s nimi jazyk SQL. Ten je rozdělen do čtyř různých jazyků podle funkcionality.

Jazyk DDL (Data Definition Language) slouží pro definici dat. Patří sem příkazy pro vytvoření databáze, tabulky či pohledu, změnu struktury a vymazání tabulky či pohledu. Každý příkaz tohoto jazyka zároveň funguje jako potvrzení všech předešlých transakcí.

Dále existuje jazyk DML (Data Manipulation Language), který je používán pro manipulaci s daty. Například pro vkládání či mazání záznamů, úpravu vložených záznamů či pro tvorbu dotazů.

DCL (Data Control Language) obsahuje příkazy pro udělení či odebrání práv.

Jazyk TCL (Transaction Control Language) se používá pro řízení transakcí.

Existuje několik základních příkazů. Obecně jsou to:

DDL

`CREATE`
- vytváří databázové objekty (tabulky, pohledy,…)

`DROP`
- odstraňuje databázové objekty

`ALTER`
- změní definici atributů v tabulce, umožňuje přejmenovat tabulku, změnit integritní omezení či jiné vlastnosti (například AUTO_INCREMENT) a přidat či odebrat atributy

– tento příkaz je dobré používat jen tehdy, je-li změna prováděna na prázdné tabulce, tedy na takové, která ještě není naplněna daty

DML

`INSERT` – vkládá záznamy do tabulky

`UPDATE` – mění data v tabulce

`SELECT` – používá se při tvorbě dotazů – vrací výsledky dotazu

`DELETE` – smaže/odstraňuje záznamy z tabulky

DCL

`GRANT` – používá se pro přidělení práv

`REVOKE` – odebrání práv

TCL

`COMMIT` – potvrzení transakce

`ROLLBACK` – zrušení transakce

Pro lepší představu si jednotlivé příkazy rozebereme podrobněji. Nejprve je však třeba uvést několik pravidel. Co se týká samotných příkazů, MySQL není v tomto ohledu tzv. case-sensitive. To znamená, že nezáleží na tom, zda píšeme příkaz velkými či malými písmeny. Například příkaz na vytvoření tabulky `CREATE TABLE` lze napsat jako `create table` či `Create Table` apod.

Názvy tabulek i atributů píšeme raději bez diakritiky. Mohl by být problém s nastavením správného kódování a názvy by se nezobrazovaly správně. Objekty databáze jsou na rozdíl od příkazů case-sensitive.

Příkazy můžeme rozložit do více řádků. Je důležité, že na konci každého příkazu musí být středník.

```
CREATE DATABASE jmeno_databaze;
```

– slouží k vytvoření databáze

```
CREATE TABLE zamestnanci(
Cislo_zamestnance BIGINT PRIMARY KEY AUTO_INCREMENT,
Jmeno VARCHAR (20) NOT NULL,
Prijmeni VARCHAR(40) NOT NULL
);
```

- vytvoření tabulky zaměstnanců, která bude obsahovat číslo zaměstnance, přidělené firmou, i jeho jméno a příjmení
- vždy nejprve definujeme název sloupce (atribut) a poté jeho datový typ či jiná integritní omezení
- jednotlivé atributy jsou odděleny čárkou, integritní omezení pouze mezerou

```
DROP TABLE zamestnanci;
```

- smaže tabulku zaměstnanců a s ní všechny hodnoty, které do ní byly uloženy

```
DROP DATABASE jmeno_databaze;
```

- smaže konkrétní databázi

```
INSERT INTO
zamestnanci (Cislo_Zamestnance, Jmeno, Prijmeni)
VALUES (1, 'Josef', 'Novák');
```

- vloží do tabulky zaměstnanců nový záznam – zaměstnance Josefa Nováka, který má číslo 1
- je dobré si zapamatovat, že hodnoty pro řetězcové typy musí být vkládány mezi apostrofy
- tento příkaz lze vložit i ve zkrácené podobě:

```
INSERT INTO zamestnanci VALUES (1, 'Josef', 'Novák');
```

- pokud však použijeme zkrácenou podobu, musíme vyplnit hodnoty všech atributů a to ve správném pořadí

```
DELETE FROM zamestnanci WHERE Jmeno = 'Josef';
```

- vymaže všechny záznamy z tabulky o zaměstnancích, kteří se jmenují Josef
- do tohoto příkazu lze dát libovolnou podmínku, nemusí se nutně jednat o jméno

– kdybychom nezadali podmínku, dojde k vymazání všech záznamů v tabulce

```
UPDATE zamestnanci SET Jmeno = 'Jan'
WHERE Prijmeni = 'Novák';
```

– změní jméno zaměstnance, jehož příjmení je Novák, na Jan
– podmínka opět může být jiná

```
ALTER TABLE zamestnanci
ADD (Rodne_cislo VARCHAR(11) NOT NULL);
```

– příkaz `ALTER` slouží pro změnu struktury již existující tabulky (lze přidávat či mazat sloupce, měnit definici sloupců či přejmenovávat tabulky)
– tento příkaz přidá do tabulky zaměstnanců atribut obsahující rodná čísla zaměstnanců

```
ALTER TABLE zamestnanci DROP Rodne_cislo;
```

– odebere z tabulky zaměstnanců atribut s rodnými čísly

```
ALTER TABLE zamestnanci RENAME pracovnici;
```

– přejmenuje tabulku `zamestnanci` na `pracovnici`

```
ALTER TABLE zamestnanci
CHANGE Cislo_zamestnance VARCHAR(10);
```

– změní datový typ atributu obsahujícího čísla zaměstnanců na `VARCHAR(10)`

```
ALTER TABLE zamestnanci
ADD CONSTRAINT Zamestnanec_pk PRIMARY KEY (Zamestnanec_ID);
```

– přiřadí atributu `Zamestnanec_ID` vlastnost primárního klíče

```
ALTER TABLE zamestnanci
DROP CONSTRAINT Zamestnanec_pk;
```

– odstraní primární klíč `Zamestnanec_pk` z tabulky `zamestnanci`

Inspiraci na uvedené základní příkazy jsem nalezla v odborné literatuře (viz zdroj [1]).

2.6 Kódování

Systém MySQL dovolí ukládat řetězcové typy v různých znakových sadách. Ty mohou být specifikovány nejen pro celou databázi ale i zvlášť pro jednotlivé tabulky či dokonce atributy. Nejlépe je však používat jednu znakovou sadu pro celou databázi.[9]

V programu MySQL Workbench najdeme kódování pod názvem *Collation*. Pro svou práci jsem zvolila `utf8 – default collation`.

2.7 Typy tabulek – enginy

Protože velmi záleží na vnitřní reprezentaci dat v databázi, vzniklo několik typů tabulek. Každý z nich má své výhody i nevýhody.

U některých důležitých změn v databázi, musíme zajistit, že se množina dotazů vykoná buď celá, nebo se nestane nic. Nesmí se v průběhu vykonávání příkazů stát, že dojde k chybě a nebude vše dokončeno. Řešení nabízí transakce, což je vlastně posloupnost instrukcí, se kterými je nutné zacházet jako s nedělitelným celkem. Transakce vždy zahájíme, vykonáme příkazy a celou transakci potvrdíme příkazem `COMMIT` či zrušíme příkazem `ROLLBACK`. Ostatní uživatelé neuvidí změnu až do potvrzení transakce.

Existují dva transakčně bezpečné typy tabulek (InnoDB, BDB), ostatní už transakčně bezpečné nejsou (ISAM, MyISAM, MERGE, HEAP). Nejčastěji se používají enginy InnoDB a MyISAM.[1, 6, 8]

2.7.1 ISAM

ISAM (Indexed Sequential Access Method) v překladu znamená metoda indexovaného sekvenčního přístupu. Tento typ tabulek je dnes považován za zastaralý, v MySQL je k dispozici pouze do verze 4. Indexy k tabulkám jsou v tomto enginu ukládány nekomprimovaně.

Typ ISAM ukládá datové soubory s příponou .ISD a soubor indexu s příponou .ISM. Nevýhodou je, že nejsou přenositelné.

2.7.2 MyISAM

Tabulky MyISAM jsou nástupcem typu ISAM. Jejich velkou nevýhodou je to, že nepodporují transakce. Hlavní rozdíl oproti ISAM je komprimace indexů. Tabulky typu MyISAM komprimují indexy, tudíž zabírají méně místa na disku. Komprimace sice více zatěžuje procesor, ale při výkonech dnešních procesorů to není velký problém. Dalším rozdílem je také přenositelnost – tabulky MyISAM jsou přenositelné.

Při vkládání záznamů je uzamčena celá tabulka, tj. v daný okamžik nelze na tabulce vykonávat žádné další příkazy.

Datové soubory jsou ukládány s příponou .MYD a indexy s příponou .MYI. Velikost souborů je v tomto případě omezena souborovým systémem. To lze obejít pomocí tabulek MERGE (viz 2.7.3).

Existují tři podtypy MyISAM: statické, dynamické a komprimované.

Statické tabulky mají pevnou délku. Pro každý záznam je vyhrazen stanovený prostor. Tento prostor patří danému záznamu i v případě, že zůstane nevyužit. Výhodou tohoto mechanismu je rychlost a snadná rekonstrukce po selhání. Je to dáno tím, že pozice záznamů jsou pevně dané a MySQL tedy přesně ví, kde se jednotlivé záznamy nacházejí. Velkou nevýhodou je požadování většího diskového prostoru.

Dynamické tabulky mají oproti statickým různé délky. Ukládaná data si vyžádají jen tolik prostoru, kolik opravdu potřebují. Každý záznam má hlavičku, která obsahuje informace o jeho délce a rozmístění. Tento typ tabulek sice zabírá méně diskového prostoru, zato však vyžaduje pravidelnou údržbu kvůli možnosti fragmentace. Kdybychom například chtěli změnit jméno zaměstnance Nového na Novotný, bylo by nové příjmení rozděleno na více částí. Pro příjmení tohoto zaměstnance byly totiž původně vyhrazeny jen čtyři znaky, zbytek by se musel uložit na jiné místo.

Komprimované tabulky jsou určeny pouze ke čtení. Vyžadují méně diskového prostoru. Jsou vhodné pro taková data, která se nebudou měnit.

2.7.3 MERGE

Tabulka MERGE je virtuální tabulka nad dvěma či více tabulkami MyISAM, které jsou identické (stejný název, typ, stejné atributy,…). Tento engine dovoluje rozdělit příliš velkou tabulku (tedy tabulku s velkým množstvím záznamů) na několik menších částí, které lze umístit na odlišné disky.

2.7.4 HEAP

Jedná se o nejrychlejší typ tabulek, protože data se ukládají jen do paměti. Z toho plyne jejich velká nevýhoda, a sice, že při pádu systému můžeme o data přijít. Dalším omezením je množství ukládaných dat, které závisí na velikosti paměti. Jako omezení můžeme označit i to, že nepodporují vlastnost `AUTO_INCREMENT` ani sloupce typu `BLOB` či `TEXT`.

2.7.5 InnoDB

Tabulky InnoDB jsou transakčně bezpečné. Používají uzamykání na úrovni řádků, což znamená, že pro uživatele je v daném okamžiku nedostupný pouze upravovaný řádek, nikoli celá tabulka. Tento typ je vhodný tehdy, používáme-li častěji příkazy pro změnu záznamů (`UPDATE, INSERT`) než pro výběr záznamů (`SELECT`).

Od tabulek MyISAM se liší v tom, že všechny tabulky a indexy nejsou uloženy v nějakém adresáři, ale v prostoru tabulek, který je navenek reprezentován jako jediný soubor (typicky *ibdata1*).

Další odlišnost se týká cizích klíčů. MyISAM sice dovolí definovat atribut, jenž by měl sloužit jako cizí klíč, ale nerespektuje ho. Po naplnění tabulky nebudeme mít kontrolu, že daný záznam existuje.

2.7.6 BDB

Tento typ byl vyvinut na University of California ve městě Berkeley. Tabulky BDB (Berkeley Database) jsou podobné typu InnoDB. V současné době je rozhraní mezi MySQL a BDB, které existuje nezávisle na MySQL, stále ve fázi vývoje. Není úplně spolehlivé a vyskytují se v něm zvláštní chyby. Proto je třeba zvýšené opatrnosti při použití tohoto typu tabulek.

Informace v kapitole 2.7 byly čerpány ze zdrojů [1], [6], a [7].

3 MySQL Workbench

MySQL Workbench je jednotný vizuální nástroj pro databázové architekty, vývojáře a databázové administrátory. S jeho pomocí lze nejen modelovat data, ale i konfigurovat a spravovat server, provádět administraci uživatelů, zálohovat apod.

MySQL Workbench funguje na operačních systémech Windows, Linux a Mac OS X.

V příloze je popsáno, jak nainstalovat databázový systém MySQL i program MySQL Workbench. Praktická část této bakalářské práce byla vypracována nástrojem MySQL Workbench ve verzi 6.0 na operačním systému Microsoft Windows 8.

3.1 Modelování dat

Pomocí nástroje MySQL Workbench lze vytvářet logické (neboli relační) či fyzické datové modely. Rozdíl mezi nimi tkví v tom, že objekty fyzického datového modelu jsou na rozdíl od logického již fyzicky uloženy v databázi.

V MySQL Workbench existuje několik způsobů, jak takový logický model vytvořit. Jednotlivé cesty jsou popsány v kapitole 5. Nástroj nabízí možnost zakládat různé databázové objekty, například tabulky, pohledy atd. U tabulek definujeme jejich názvy, atributy, datové typy, kódování, integritní omezení a další různé vlastnosti. Názvy tabulek by neměly obsahovat diakritiku ani mezery, stejně tak jednotlivé atributy. Ke každému atributu vybereme vhodný datový typ z nabídky.

Atributům můžeme nastavit tyto vlastnosti:

PK	– PRIMARY KEY – atribut je součástí primárního klíče
NN	– NOT NULL – v takto označeném sloupci musí být pro každý záznam vždy uvedena nějaká hodnota
UQ	– UNIQUE – jednotlivé hodnoty tohoto atributu musí být jedinečné
BIN	– BINARY – takto označený atribut bude obsahovat pouze hodnoty v binární podobě (binární data)
UN	– UNSIGNED – povoluje pouze nezáporné číselné hodnoty
ZF	– ZERO FILL – hodnoty ve sloupci s touto vlastností budou zobrazovány s určitým počtem nul na začátku tak, aby byla vždy zobrazena celá definovaná šířka (například u datového typu INT(4) by bylo číslo 1 zobrazeno jako 0001)
AI	– AUTO_INCREMENT – takto označený sloupec si bude automaticky generovat unikátní číselné hodnoty

Nástroj MySQL Workbench nabízí i několik šablon, které lze použít jako základ pro tabulky a pak je libovolně upravit.

Pohledy musíme vytvořit pomocí jazyka SQL.

S tvorbou logického modelu souvisí EER diagram. Jde vlastně o grafické znázornění modelu, většinou tedy tabulek a vazeb mezi nimi. MySQL Workbench umožňuje

vytvořit přehledný diagram i pro složitá schémata. Jednotlivé části diagramu (například skupiny tabulek) lze barevně označit či jim přidat popisek formou textového pole. Je zde i možnost libovolně měnit nejen typ a velkost písma, ale dokonce i vzhled tabulek a relací. Výsledný diagram můžeme uložit jako obrázek či do formátu pdf. Také lze určit výšku a šířku ukládaného modelu, kde jednotkami jsou stránky. Samozřejmostí je i volba tisku.

Pro tvorbu fyzického datového modelu se nejprve musíme přihlásit do databáze. Vytvoření připojení a uživatelských účtů je popsáno v příloze (viz kapitola 4 Přílohy B).

Dvojitým poklikáním na nějaké připojení a následným zadáním správného hesla se připojíme k databázi. Pak můžeme vytvářet různé databázové objekty. Postupům pro vytváření tabulek a relací je věnována kapitola 6. Hotové tabulky pak lze naplnit daty.

3.2 Administrace uživatelů, správa serveru

Pomocí nástroje MySQL Workbench lze vytvářet uživatelské účty a přiřazovat jim nějaká práva (postup viz kapitola 4 Přílohy B). Systém také poskytuje informace o tom, kdy se jednotliví uživatelé připojili do databáze.

3.3 Zálohování

Po připojení k databázi klikneme v poli *Navigator* na *Data Export*. Zvolíme, kterou databázi chceme zálohovat a vhodně ji pojmenujeme. Tím je vytvořen soubor s koncovkou .sql, který představuje zálohu.

3.4 Nápověda

Program nabízí standardní nápovědu, ve které stačí buď zadat hledaný výraz, nebo najít vysvětlení či postup v konkrétní kapitole nápovědy.

Další nabízenou možností je propojení s webovými stránkami www.mysql.com, kde lze vyhledat potřebnou pomoc.

4 Řešený relační model

Pod pojmem relační model si představme jeden ze způsobů ukládání dat a vazeb mezi nimi. Data jsou ukládána do tabulek, které se skládají z řádků a sloupců.[2] Další způsoby ukládání dat zde rozebírat nebudu, protože systém MySQL je nevyužívá.

4.1 Představení řešeného modelu

Pro svou práci jsem si vybrala zjednodušený systém pošty ve městě (model viz Příloha A). Model tohoto systému se skládá z třinácti tabulek. Po naplnění bude obsahovat různé informace od údajů o zaměstnancích pošty po jednotlivé adresy daného města.

Předpokládejme, že ve městě je několik poboček pošty. Jednotlivé pobočky mohou být listovní, balíkové či motorizované, tj. kombinace dvou prvních. Město je rozděleno na několik okrsků (částí), přičemž v každém okrsku je nějaká listovní a někde i balíková pobočka. Každá pobočka má různé zaměstnance na různých pozicích. Každý zaměstnanec na pozici doručovatele roznáší listovní zásilky do jednoho okrsku. Okrsky, ve kterých není balíková pobočka, mají přiřazenou takovou pobočku v jiném okrsku. Jednotlivé části města jsou identifikovány adresami, na kterých žijí klienti (obyvatelé). Každý klient může být adresátem či odesílatelem, tj. zásilky přijímá či odesílá. Pošta zajišťuje doručování zásilek.

4.2 Rozbor modelu

4.2.1 Tabulky

Model obsahuje nejen tabulky, ale i několik číselníků. Číselník je zvláštním druhem tabulky, protože obsahuje pouze výčet hodnot.

Číselníky:

`druhy_pobocek` – všechny existující druhy poboček (balíková, listovní, motorizovaná)

`druhy_zasilek` – např. balík do ruky, doporučený dopis, atd.

`pozice` – obsahuje informace o různých pozicích, náplni práce i hodinové mzdě

Tabulky:

`okrsky` – obsahuje základní informace o jednotlivých okrscích, na které je město rozděleno

`zamestnanci` – seznam všech zaměstnanců, jejich osobní údaje, pozice, informace o tom, pro jakou pobočku pracují

`pobocky` – seznam poboček pro dané město, jejich sídlo atd.

`pobocky_pro_okrsky` – pobočky, které neleží na území daného okrsku, ale poskytují mu služby

`pobocky_v_okrscich` – pobočky, které leží na území daného okrsku a poskytují mu služby

`adresy` – soupis všech adres ve městě

`klienti` – seznam všech klientů (obyvatel), kteří v dané obci žijí

`dorucene_zasilky` – informace o balících či listovních zásilkách, které je třeba doručit obyvatelům jiného města

`poslane_zasilky` – informace o balících či listovních zásilkách, které byly odeslány jakýmkoli obyvatelem daného města

`dorucovani` – z této tabulky lze zjistit, které doručovatele má konkrétní okrsek

4.2.2 Relace

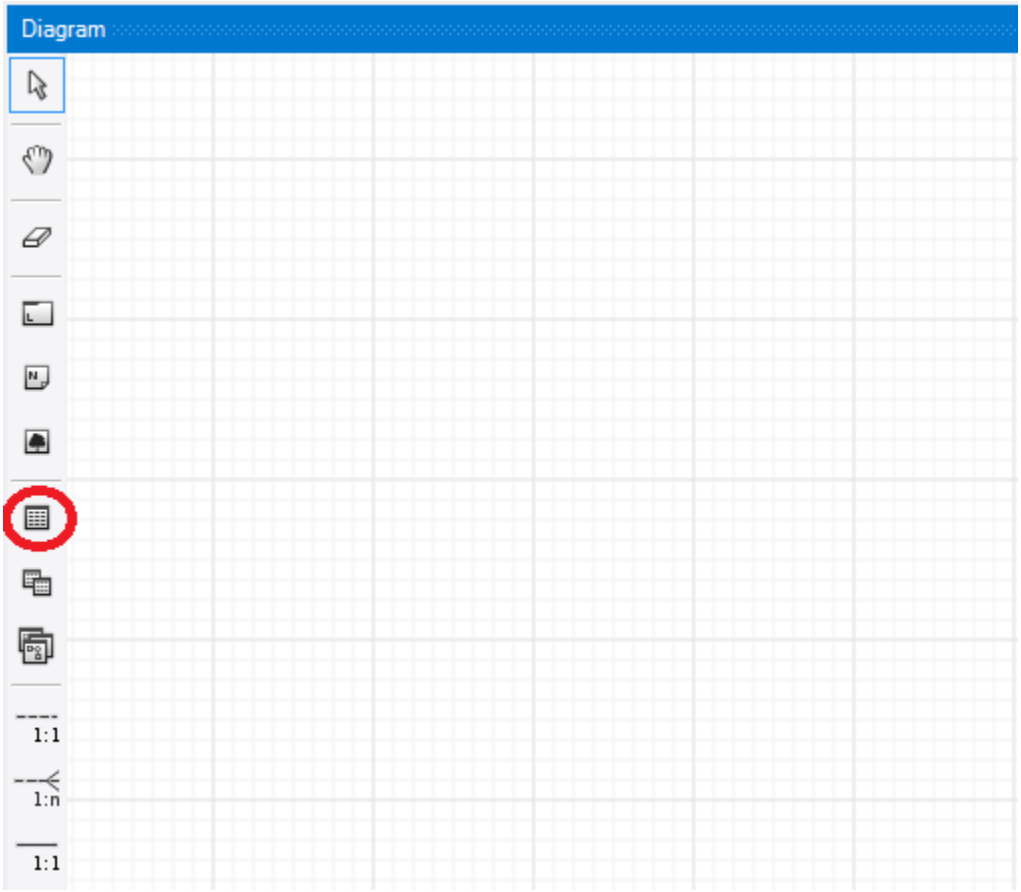
Všechny relace jsou typu 1:N. Tabulky `pobocky_pro_okrsky` a `pobocky_v_okrscich` vznikly jako důsledek rozkladu vazby M:N mezi tabulkami `pobocky` a `okrsky`, protože v každém okrsku je jedna či více poboček a zároveň každá pobočka může doručovat více okrskům.

5 Realizace logického datového modelu

Před vytvářením fyzického datového modelu je vhodné vytvořit logický datový model. Snáze si tak uvědomíme nedostatky či chyby ve schématu. Návrh pak můžeme použít jako základ pro fyzický datový model.

5.1 Vytváření objektů v poli

1) Nejdříve spustíme program MySQL Workbench.
2) Klikneme na ikonku malého plus (v dolní části obrazovky vlevo).
3) Otevře se záložka pro vytvoření modelu.
4) Poté poklikáme na ikonu *Add Diagram* a otevře se záložka s polem, kam budeme umisťovat tabulky.
5) Po levé straně tohoto pole je nabídka objektů, které lze vytvořit. Najdeme tam nejen tabulky, ale i pohledy, relace či textová pole. Klikneme na ikonku tabulky (viz Obrázek 3) a pak klikneme na libovolné místo v poli.
6) Tabulku pak musíme upravit, především definovat její název, atributy, atd. Tabulku v diagramu vybereme (tedy na ní dvakrát poklikáme), tím se zobrazí okno s vlastnostmi tabulky.
7) Do příslušného pole (viz Obrázek 4) pak nejdříve vyplníme název tabulky. Je dobré připomenout, že název by neměl obsahovat mezery ani diakritiku.
8) Dále definujeme atributy. Do sloupce *Column Name* (viz Obrázek 4) píšeme názvy atributů a ve sloupci *Datatype* (viz Obrázek 4) volíme vhodné datové typy.
9) V neposlední řadě musíme také definovat integritní omezení. Tím je myšleno nastavení primárního klíče, nenulových atributů atd. (viz kapitola 2.3). Nemusíme však definovat pouze integritní omezení. Nástroj MySQL Workbench nabízí i jiné vlastnosti, které lze atributům přiřadit (viz kapitola 3.1). Ve sloupci *Default* pak můžeme zvolit výchozí hodnotu pro atributy.



Obrázek 3 - Pole pro vytváření databázových objektů

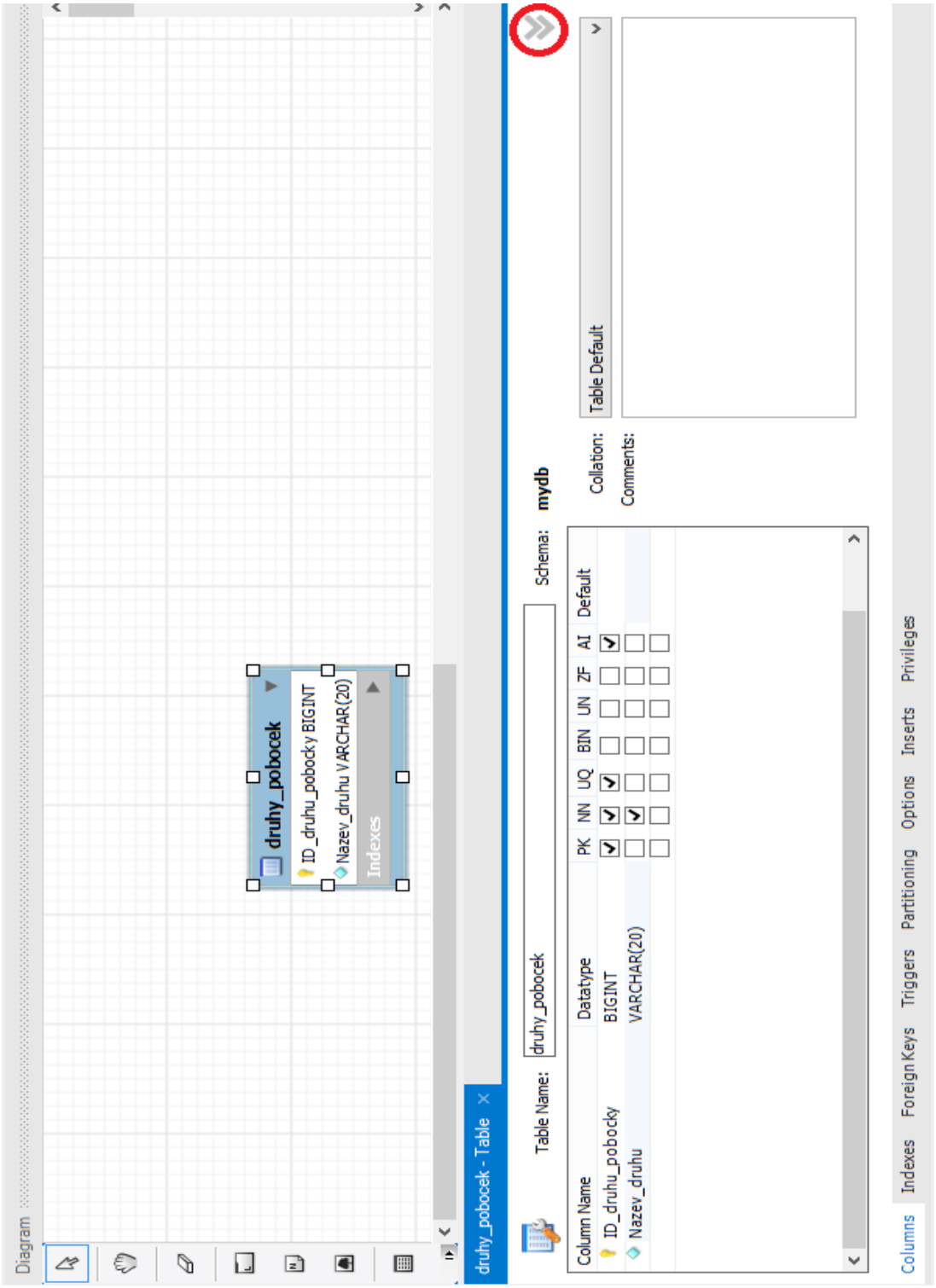
10) Můžeme si vybrat i engine, ve kterém chceme tabulku vytvořit. K tomuto nastavení se dostaneme pomocí zdvojené šipky dolů vedle pole s názvem *Schema* (viz Obrázek 4). Výběr enginu závisí na možnostech, které budeme od databáze očekávat (viz kapitola 2.7).

11) Jakmile máme tabulky vytvořené, propojíme je pomocí relací.

12) Z nabídky po levé straně pole vybereme vhodnou relaci, klikneme na první tabulku a poté klikneme na další tabulku. V první tabulce se automaticky vytvoří další sloupec s cizím klíčem. Je tedy třeba si uvědomit, do které tabulky chceme přidat cizí klíč.

    Toto však neplatí pro relaci M:N. V takovém případě se nevytvoří sloupec s cizím klíčem, ale nová rozkladová tabulka.

13) Posledním krokem může být úprava vygenerovaných názvů atributů cizího klíče.



Obrázek 4 - Definování tabulky

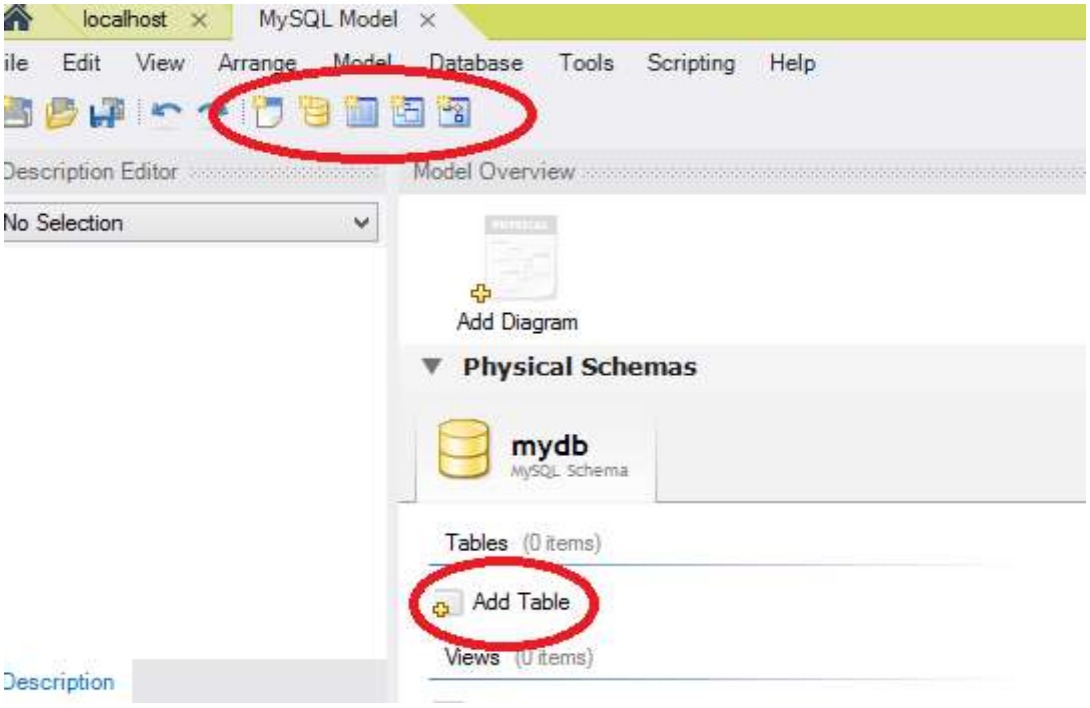
5.2 Převedení definovaných tabulek do diagramu

1) Začneme spuštěním programu MySQL Workbench a vytvořením nového modelu (viz body 1-3 v kapitole 5.1).

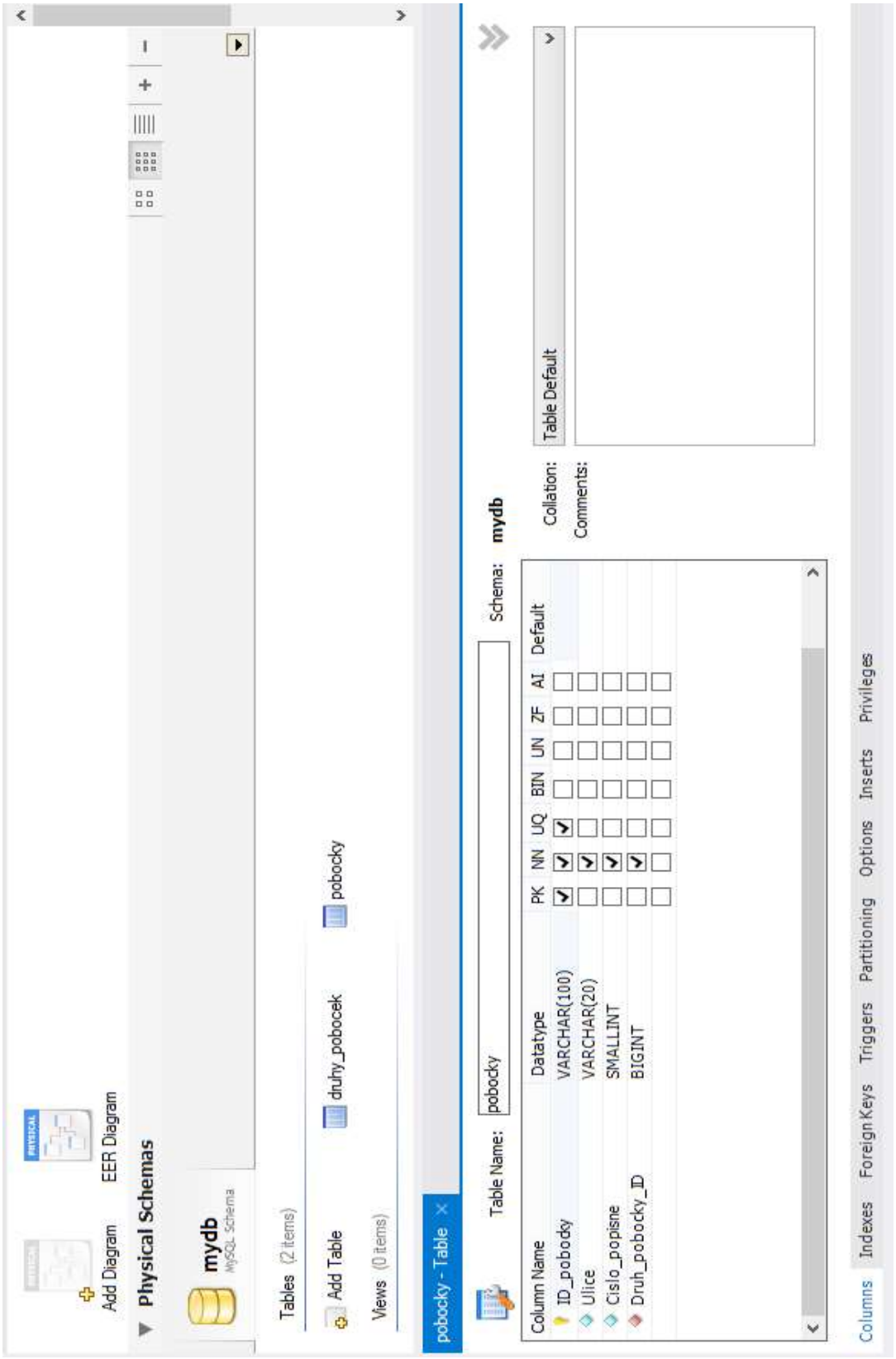
2) V záložce, kterou jsme právě otevřeli, je seznam vytvořených databázových objektů. Na začátku tam samozřejmě nemáme nic. Za zmínku jistě stojí, že lze vytvářet nejen tabulky, ale i pohledy či skupiny tabulek.

3) Databázové objekty můžeme vytvořit pomocí panelu s nabídkou v levé horní části obrazovky (viz Obrázek 5).

4) Existuje však i jiný způsob. Přibližně v polovině obrazovky máme zatím prázdný seznam tabulek s tlačítkem *Add Table* (viz Obrázek 5). Když na něj poklikáme, dojde k vytvoření tabulky. Navíc se v dolní části obrazovky otevře okno (viz Obrázek 6) pro úpravu dané tabulky. Toto okno je naprosto stejné jako u prvního způsobu vytváření tabulek (viz Obrázek 4).



Obrázek 5 - Vytváření databázových objektů



Obrázek 6 - Definování tabulky

5) V něm definujeme název, atributy, jejich vlastnosti a datové typy (viz body 7-10 kapitola 5.1).

6) Nezapomeňte, že v případě relace M:N je třeba vytvořit rozkladovou tabulku ručně.

7) Rozdíl oproti předchozí metodě tkví ve vytváření relací. Ty zde vytvoříme tak, že do tabulek přidáme navíc sloupce, které reprezentují nějaký atribut z cizí tabulky. Jako příklad uvedu tabulky `druhy_pobocek` a `pobocky`, které chci spojit relací 1:N. Do tabulky `pobocky` proto přidám atribut `druh_pobocky_ID`, který bude reprezentovat druhy jednotlivých poboček. Tento atribut, který bude sloužit jako cizí klíč, musí být stejného datového typu jako u tabulky `druhy_pobocek`, kde plní funkci primárního klíče.

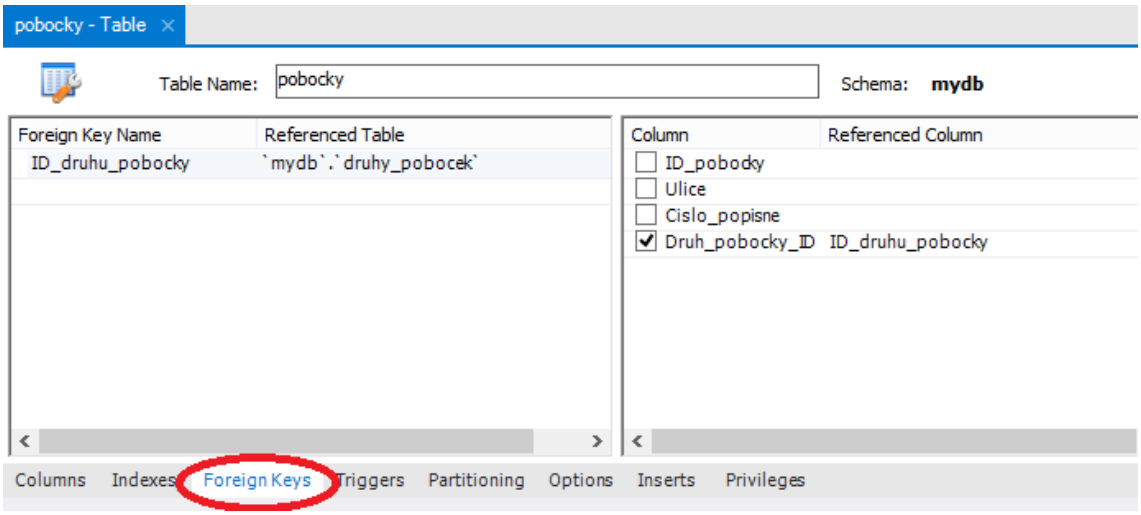
8) K předchozímu bodu doplním ještě jeden postřeh. Jestliže budu nejdříve vytvářet tabulku `pobocky`, můžu do ní přidat atribut `druh_pobocky_ID`. Nelze ho však definovat jako cizí klíč. Tabulka `druhy_pobocek` totiž ještě neexistuje a nemůžu tedy vytvářet relaci.

9) Okno, ve kterém jsme doteď definovali tabulku, obsahuje více záložek. Zatím jsme pracovali pouze v záložce *Columns* (názvy záložek najdeme v dolní části okna). Pro dokončení tvorby relací se musíme přepnout do záložky *Foreign Keys*. Tam definujeme vhodný název cizího klíče a zvolíme tabulku, se kterou chceme vytvářet vazbu (viz Obrázek 7).

10) Pak jen zaškrtneme vytvořený atribut(y), který(é) bude(ou) součástí cizího klíče, a přiřadíme mu příslušný sloupec z druhé tabulky.

11) Po vytvoření všech tabulek a jejich propojení poklikáme na ikonu *Add Diagram*.

12) Otevře se nová záložka s prázdným polem. Po levé straně je seznam vytvořených tabulek. Myší přetáhneme jednotlivé tabulky do pole a tím vytvoříme diagram.



Obrázek 7 - Definice relací/vazeb mezi tabulkami

V diagramu uvidíme nejen tabulky, ale i vazby mezi nimi, tedy jednotlivé relace. Nástroj MySQL Workbench neumí umístit tabulky do pole tak, aby se relace nekřížily. Je proto třeba jednotlivé objekty vhodně rozmístit tak, aby byl diagram přehledný.

5.3 Rozdíly mezi uvedenými postupy

Pokud neuděláme chybu v definování jednotlivých tabulek či relací, výsledné modely se téměř nebudou lišit. Jediným rozdílem jsou názvy cizích klíčů, které se v případě prvního způsobu (viz kapitola 5.1) generují automaticky, zatímco v druhém způsobu (viz kapitola 5.2) je definujeme sami. Také názvy rozkladových tabulek budou různé.

Z mého pohledu je pohodlnější umisťování tabulek do pole. Vytvoření relací a potažmo cizích klíčů je tímto způsobem jednodušší a umožní předejít různým chybám.

5.4 Vytvoření skriptu *.sql

Ať už jsme logický datový model vytvořili prvním či druhým způsobem, získáme z něj skript s koncovkou .sql takto:

1) V záložce *File* vybereme možnost *Export* a *Forward Engineer SQL CREATE Script …*
2) V horním okně vybereme adresář, kam skript umístíme. Také ho nějak vhodně nazveme.
3) Pokud chceme, můžeme zaškrtnout nějaké z nabízených možností. Doporučuji zaškrtnout možnost *Omit Schema Qualifier in Object Names*. Kdybychom to neudělali, vytvořené a exportované objekty by ve svých názvech obsahovaly i název databáze, ve které byly vytvořeny. Tím by vznikl problém při pokusu o import do databáze s jiným jménem.
4) Klikneme na tlačítko *Next* a zkontrolujeme, zda se do skriptu ukládá opravdu všechno. Jestliže je vše v pořádku, opět klikneme na *Next*.
5) Ukáže se nám vygenerovaný skript v jazyce SQL.
6) Pak už stačí jen dokončit proces vytvoření skriptu kliknutím na tlačítko *Finish*.

6 Realizace fyzického datového modelu

6.1 Import souboru s koncovkou *.sql

1) Nejprve spustíme program MySQL Workbench.

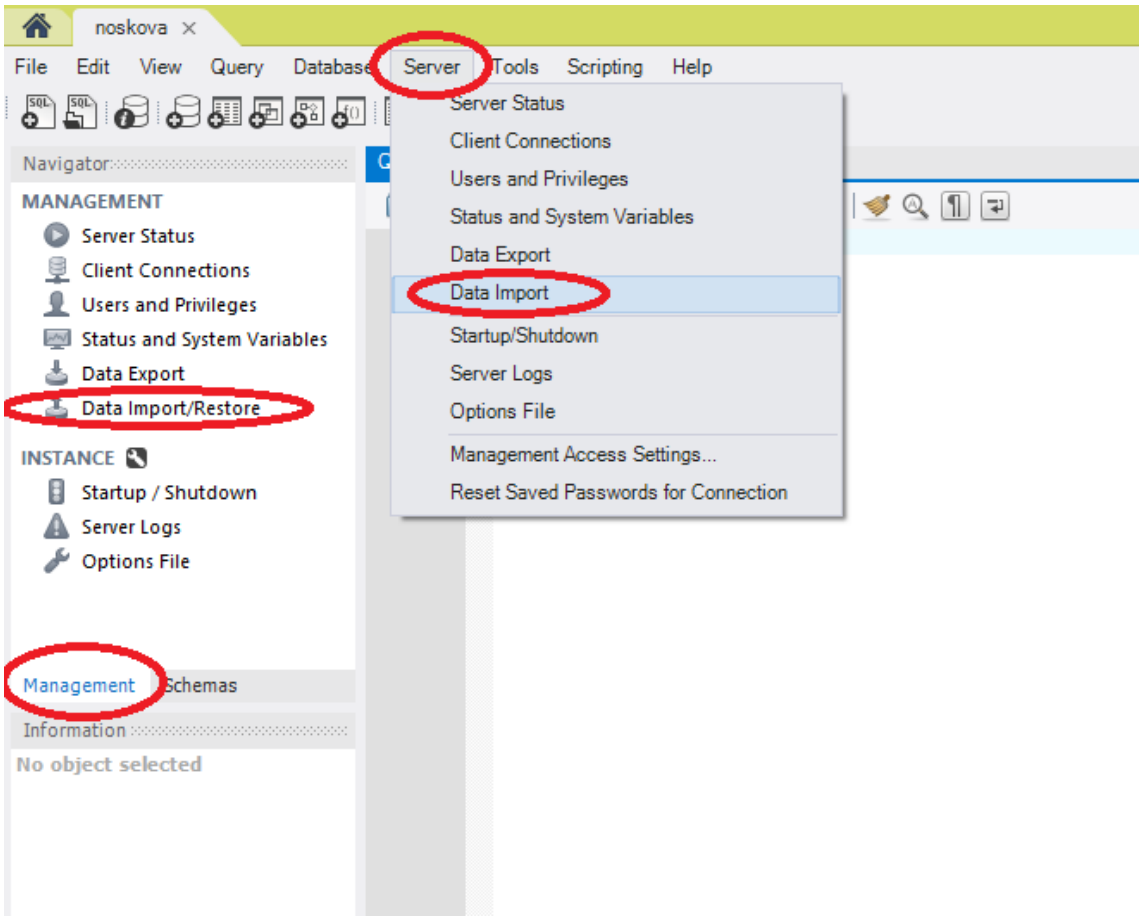
2) Poté je třeba se přihlásit do databáze jako běžný uživatel. Poklikáme na vybrané připojení a zadáme požadované heslo.

   Vytváření uživatelských účtů a databází je podrobněji rozebráno v příloze (viz Příloha B).

3) Po přihlášení máme 2 možnosti (viz Obrázek 8):

a. Klikneme na *Server -> Data Import*.

b. V záložce *Management* klikneme na *Data Import/Restore*.



Obrázek 8 - Import dat

Obrázek 9 - Definice importovaných dat

4) Systém bude možná opět požadovat heslo. Poté se otevře nová záložka věnovaná nastavení importu dat (viz Obrázek 9).

5) Zvolíme možnost *Import from Self-Contained File*.

6) Hned vedle této volby je pole, obsahující cestu k souboru s vytvořenými tabulkami z logického modelu, který chceme importovat. Vybereme tedy vhodný soubor, například ten, který jsme sami vygenerovali (viz kapitola 5.4).

7) Také musíme vybrat databázi, do které chceme importovat tabulky. Já jsem zvolila databázi s názvem `noskova`. K tomu slouží pole s názvem *Default Target Schema*. Pokud máme k tomu určená práva, můžeme založit novou databázi.

8) Klikneme na tlačítko *Start Import*, zadáme heslo a pak už se jen vygenerují tabulky.

9) Ty nyní můžeme naplnit daty (viz kapitola 7).

6.2 Tabulky vygenerované z logického modelu

1) Začínáme v okamžiku, kdy máme hotový logický datový model, tedy vytvořeny všechny tabulky, které jsou propojené relacemi. Nacházíme se v záložce pro tvorbu logického modelu (viz Obrázek 5) nebo v záložce s polem pro tvorbu diagramu (viz Obrázek 3).

2) Zvolíme možnost *Database* a poté *Forward Engineer ...*

3) Zobrazí se okno (viz Obrázek 10), ve kterém definujeme, k jakému připojení a do jaké databáze bude model generován. Protože připojení jsou již vytvořena, mají přiřazeny konkrétní databáze a nelze je v tomto okamžiku měnit.

4) Po výběru libovolného připojení klikneme na tlačítko *Next*.

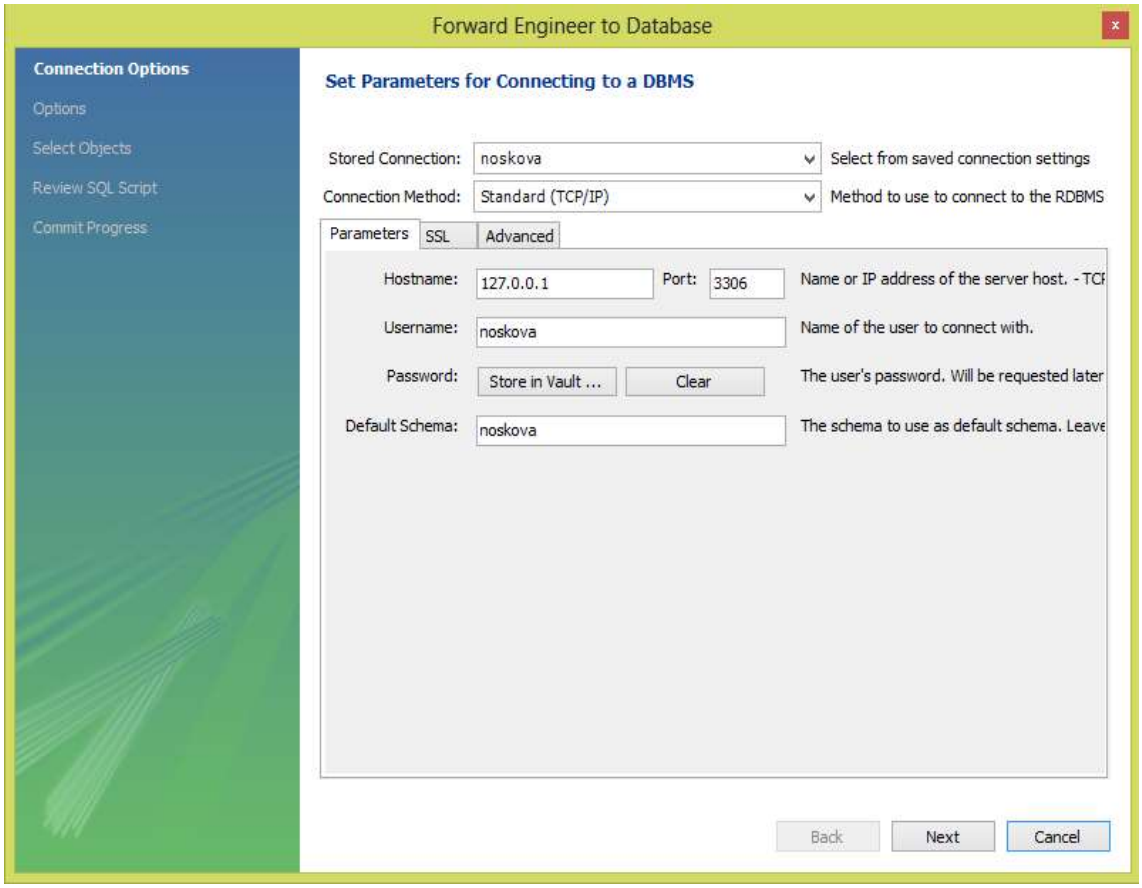
5) Můžeme opět vybrat nějaké možnosti (viz bod 3, kapitola 5.4).

6) Opět klikneme na tlačítko *Next*. Systém po nás může vyžadovat zadání hesla k vybranému připojení.

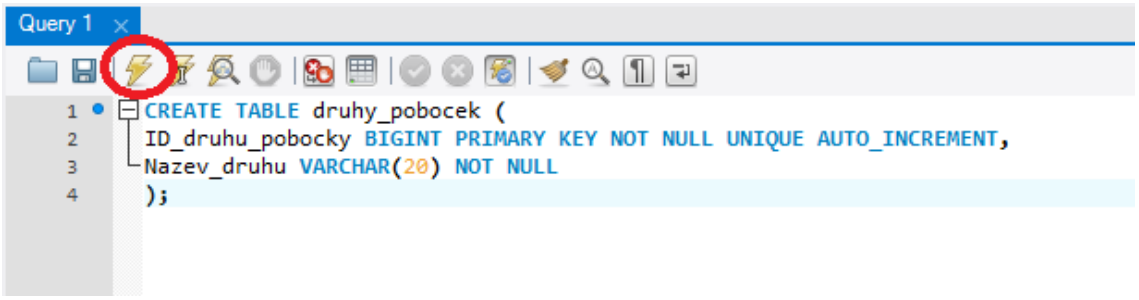
7) Zkontrolujeme, zda se budou přenášet všechny databázové objekty, které chceme. Jestliže je vše v pořádku, klikneme na *Next*.

8) Následuje kód v jazyce SQL, pro vytvoření daných objektů do databáze. Opět klikneme na *Next*.

9) Provedou se zvolené operace. Pokud vše proběhlo v pořádku, můžeme okno zavřít.



Obrázek 10 - Forward Engineer



Obrázek 11 - SQL příkaz pro vytvoření tabulky

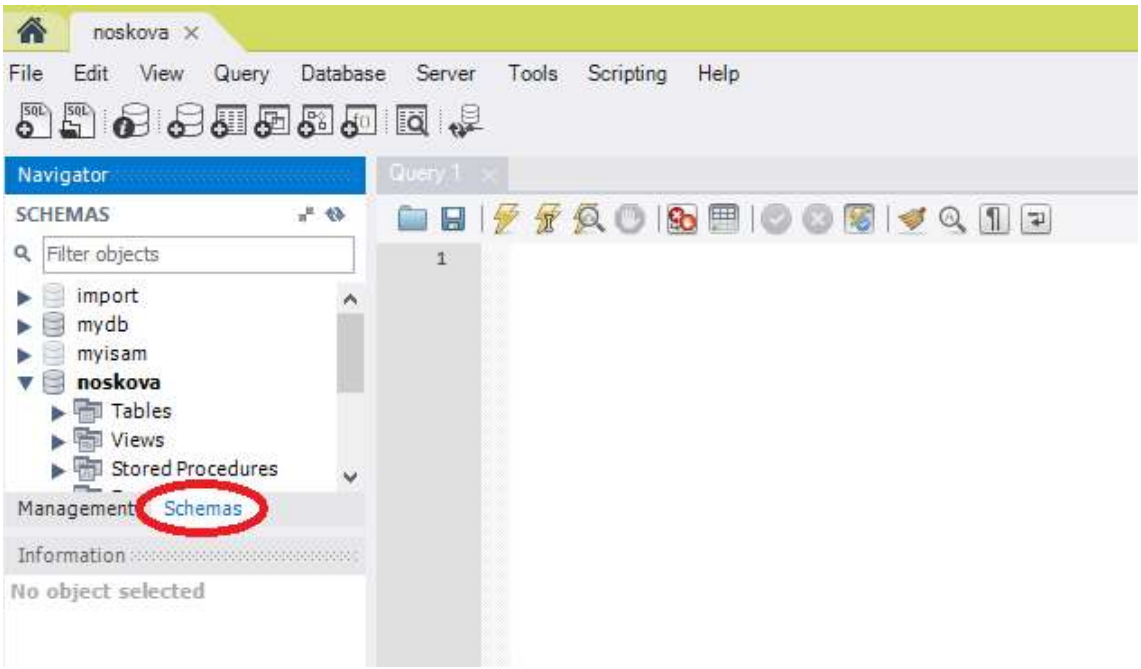
6.3 Tabulky vytvořené pomocí jazyka SQL

1) Opět začneme tím, že spustíme program MySQL Workbench a připojíme se jako běžný uživatel.
2) Po přihlášení se otevře prázdný list, na který můžeme psát příkazy v jazyce SQL.
3) Pomocí příkazů jazyka SQL pak vytvoříme tabulky a propojíme je relacemi (základní příkazy jazyka SQL jsou popsány v kapitole 2.5).
4) K tomu, aby se jakýkoli příkaz provedl, je třeba kliknout na ikonu blesku (viz Obrázek 11).

6.4 Model vytvořený nezávisle na logickém datovém modelu

1) Nejprve spustíme program MySQL Workbench.
2) Poté je třeba se přihlásit do databáze jako běžný uživatel. Vytváření uživatelských účtů a databází je podrobněji rozebráno v příloze (viz Příloha B).
3) Po levé straně se přepneme ze záložky *Management* do záložky *Schemas* (viz Obrázek 12).
4) Rozklikneme si zvolenou databázi, pravým tlačítkem myši klikneme na *Tables* a vybereme volbu *Create table*.
5) Otevře se nová záložka, která nabízí prakticky stejné možnosti jako okno pro definování tabulek u logického modelu. Je zde však rozdíl, neboť nyní se tabulky opravdu ukládají do databáze, ne jen do souboru. Do pole *Table name* napíšeme název vytvářené tabulky. Doporučuji začít tabulkami, které nemají cizí klíče.
6) Je možné také zvolit engine, ve kterém chceme tabulky vytvořit. Pokud tato možnost není otevřená, dostaneme se k ní pomocí zdvojené šipky dolů vedle pole s názvem *Schema*.

   Na příkladu je tato možnost již přístupná (viz Obrázek 13). Výběr enginu závisí na možnostech, které budeme od databáze očekávat (viz kapitola 2.7).



Obrázek 12 - Nabídka schémat

7) Také můžeme zvolit libovolné kódování (viz kapitola 2.6).

8) Tabulkám dále definujeme atributy. Jejich názvy píšeme do sloupce *Column Name*. Také musíme z nabídky zvolit vhodné datové typy (viz kapitola 2.2), případně některá z nabízených integritních omezení či jiných vlastností (viz kapitola 3.1). Ve sloupci *Default* pak můžeme zvolit výchozí hodnotu pro atributy.

9) Pro jednotlivé atributy máme v dolní části okna k dispozici seznam jejich vlastností, které lze nastavit. Za povšimnutí jistě stojí například to, že lze nastavit kódování ke každému sloupci zvlášť.

10) Je dobré myslet dopředu na vazby mezi tabulkami a podle toho již definovat atributy pro cizí klíče. Nelze však propojit tabulky, z nichž jedna zatím neexistuje.

11) Do atributů tedy přidáme sloupce, které budou součástí cizího klíče.

12) Po vytvoření tabulky klikneme na *Apply*. Zobrazí se okno s kódem v jazyce SQL pro vytvoření tabulky. Poté znovu klikneme na *Apply* a tím dojde k aplikaci skriptu, tedy k fyzickému vytvoření tabulky v databázi. Nakonec klikneme na tlačítko *Finish*.

13) Nyní můžeme tabulku naplnit daty (viz kapitola 7).

Obrázek 13 - Definování tabulky a jejích atributů

Může se stát, že budeme chtít ve vytvořené tabulce něco změnit, například přidat či odebrat atribut, vytvořit cizí klíč atd. K tomu slouží příkaz `ALTER TABLE` (viz kapitola 2.5). Klikneme tedy pravým tlačítkem na tabulku, kterou chceme upravovat, a zvolíme možnost *Alter table...*. Otevře se stejná záložka jako pro vytváření databáze. Provedeme požadované změny a postupujeme stejně jako v bodě 11.

6.5 Rozdíly mezi nalezenými cestami

Pro ty, co s MySQL začínají a nemají ještě dostatečně velké znalosti jazyka SQL, nedoporučuji tvořit fyzický datový model podle kapitoly 6.3. Propojit tabulky pomocí cizích klíčů je leckdy složité i pro pokročilé uživatele.

Jestliže jste nejdříve vytvořili logický datový model, pak je výhodné jej použít pro tvorbu fyzického modelu. Tedy postupovat podle kapitol 6.1 a 6.2.

Nejlepším řešením je podle mě tvorba logického datového modelu a na jeho základě pak vygenerování fyzického modelu. Díky tomu, že můžeme tabulky i vazby vidět v grafické podobě si snáze uvědomíme jejich vzájemné propojení.

6.6 Další možnosti

Lze také importovat data a tabulky z jiného databázového systému. A to tak, že v záložce *Database* vybereme položku *Migration Wizard*.

7 Editace dat

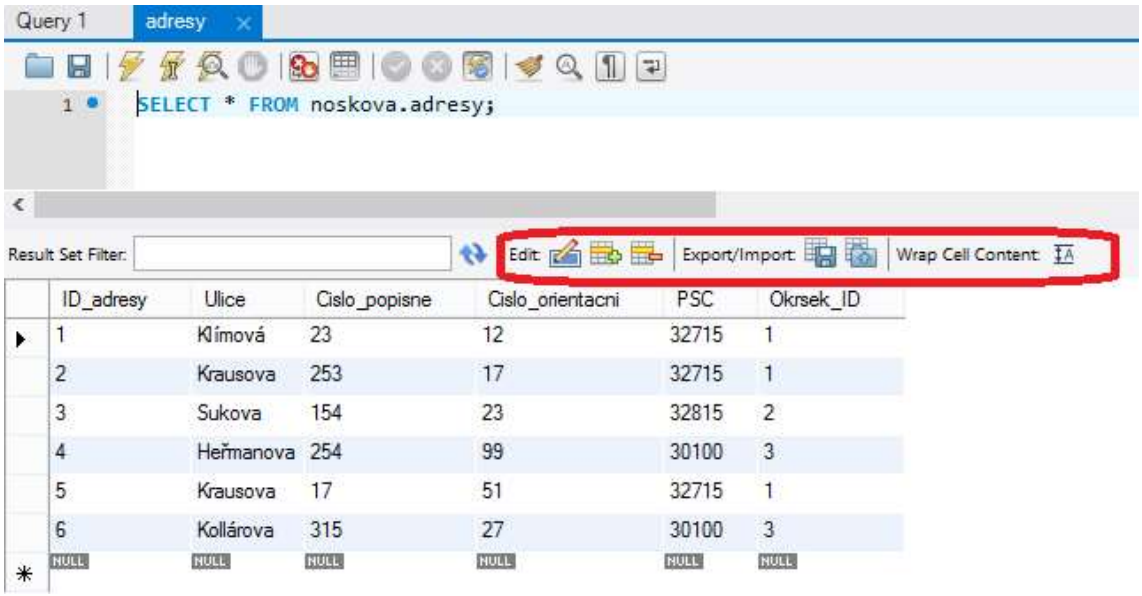
Data můžeme do tabulek vkládat či je upravovat pomocí příkazů jazyka SQL (viz kapitola 2.5). Ty píšeme do stejného pole, jako jsme vytvářeli tabulky v kapitole 6.3 (záložka *Query 1*). Příkazy vždy zrealizujeme tím, že klikneme na ikonu blesku (viz Obrázek 11). Existuje však i jiný způsob:

1) Pravým tlačítkem myši klikneme na tabulku, ve které chceme editovat data a vybereme možnost *Select Rows – Limit xxx* (Pozn.: místo xxx bude konkrétní číslo dle nastavení limitu).

2) Otevře se záložka pro editaci dat (viz Obrázek 14).

MySQL Workbench, stejně jako jiné databázové nástroje, nabízí různé možnosti manipulace s daty. Můžeme je přidávat, mazat, upravovat, dokonce i exportovat či importovat do/ze souboru.

Nakonec přidám poznámku k atributům, které mají nastavenou vlastnost `AUTO_INCREMENT`. Hodnoty v těchto sloupcích necháváme prázdné. K jejich automatickému vyplnění dojde až poté, co klikneme na tlačítko *Apply*.



Obrázek 14 - Editace dat

8 Závěr

Cílem práce bylo najít a popsat různé cesty vedoucí k vytvoření fyzického datového modelu pomocí nástroje MySQL Workbench. Nejprve však bylo třeba seznámit čtenáře se základy MySQL.

V této práci jsou popsány čtyři různé způsoby, jakými lze fyzický datový model vytvořit. Dvě z popsaných cest vycházejí z logického datového modelu, jehož tvorba je popsána v kapitole 5.

Začátečníkům v MySQL doporučuji vygenerovat fyzický model na základě logického. Díky grafickému zobrazení modelovaného systému si uživatelé snáze uvědomí propojení jednotlivých tabulek.

Uživatelé zběhlí v jazyce SQL mohou navrhovat databázi přímo pomocí příkazů SQL.

Bakalářská práce by mohla být rozšířena o podrobnější popis všech možností, které MySQL Workbench nabízí. Pak by sloužila jako uživatelská příručka tohoto databázového nástroje.

Literatura

[1] Gilfillan, I. (2003). *Myslíme v MySQL 4, knihovna programátora.* Praha: Grada Publishing, a. s.

[2] Misha. (1. březen 2014). *Úvod, Databáze*. Načteno z Databáze: http://www.databaze.chytrak.cz/

[3] *Relační databáze, Wikipedie*. (29. březen 2014). Načteno z Wikipedie otevřená encyklopedie: http://cs.wikipedia.org/wiki/Rela%C4%8Dn%C3%AD_datab%C3%A1ze

[4] *Teorie relačních databází: Integritní omezení, Manuály*. (30. březen 2014). Načteno z Manuály: http://www.manualy.net/article.php?articleID=15

[5] *MySQL: Products: Wokrbench*. (30. duben 2014). Načteno z MySQL: http://www.mysql.com/products/workbench/

[6] *Znalostní báze WEDOS, Typy tabulek*. (14. duben 2014). Načteno z Znalostní báze, WEDOS: http://kb.wedos.com/mysql/typy-tabulek.html

[7] Zajíc, P. (14. duben 2014). *LInuxsoft, Typy tabulek*. Načteno z Linuxsoft: http://www.linuxsoft.cz/article.php?id_article=968

[8] Welling, L., & Thomson, L. (2005). *MySQL Průvodce základy databázového systému.* Brno: CP Books, a. s.

[9] Zajíc, P. (21. duben 2014). *Linuxsoft: Ach, ta čeština*. Načteno z Linuxsoft: http://www.linuxsoft.cz/article.php?id_article=1091

Přílohy

Příloha A

Relační model

Program MySQL Workbench používá různé ikony pro označení vlastností a integritních omezení nastavených jednotlivým atributům. Dovoluje zvolit jeden z možných způsobů zobrazení diagramu, které má předdefinované. Lze také měnit vzhled pouze určitému druhu databázových objektů, například relací.

V ukázkovém ERA modelu jsou použity tyto:

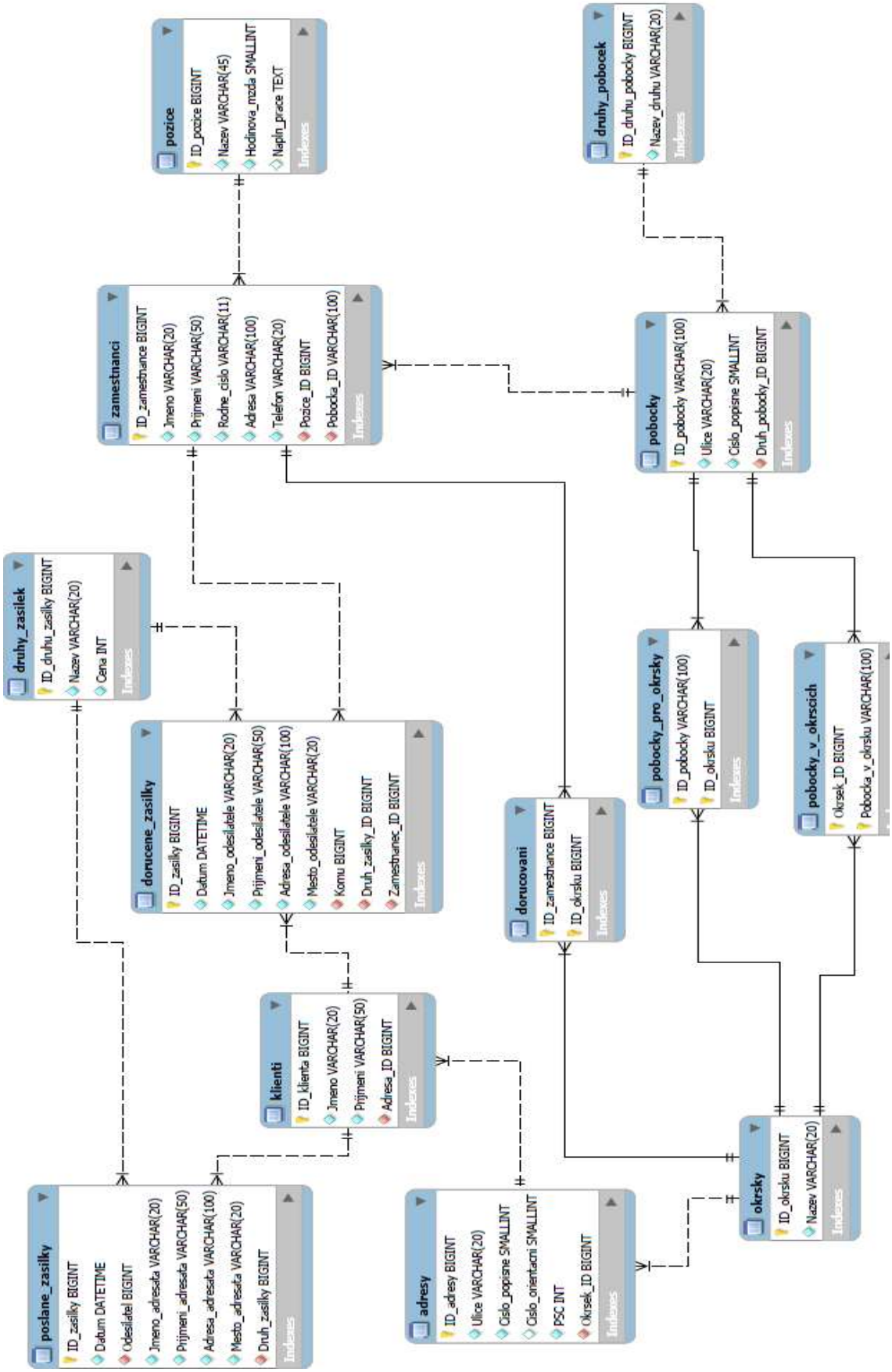
Zlatý klíč – atribut je součástí primárního klíče.

Plný kosočtverec – každý záznam musí mít u tohoto atributu vyplněnu nějakou hodnotu, tedy sloupec má nastavenou vlastnost `NOT NULL`.

Prázdný kosočtverec – záznam nemusí mít u tohoto atributu vyplněnou hodnotu (není `NOT NULL`).

Červený kosočtverec – atribut je součástí cizího klíče.

Modrý kosočtverec – atribut není součástí cizího klíče.



Obrázek 15 - ERA diagram

Příloha B

Instalace a spuštění

B.1 Instalace

Systém MySQL včetně nástroje MySQL Workbench lze zdarma stáhnout na adrese *http://dev.mysql.com/downloads/*. Budeme stahovat a instalovat nekomerční MySQL Community Edition na operační systém Windows.

Zvolíme možnost *MySQL on Windows (Installer & Tools).* Pokud již máte nainstalovaný systém MySQL, můžete stáhnout pouze MySQL Workbench. Pokud však začínáte úplně od začátku, stáhněte si oboje, tedy zvolte možnost *MySQL Installer*. K dispozici jsou vždy pouze nejnovější verze. Jestliže budete chtít stáhnout nějakou starší, musíte se přepnout do archivu (viz Obrázek 16 – odkaz *Archives*). Tam opět zvolte *MySQL Installer* a vyberte verzi, o kterou máte zájem.

Pro stažení zvolíme *mysql-installer-community*. Poté budete vyzváni k založení účtu či k přihlášení na stránkách společnosti. Chcete-li, můžete si účet zařídit. Není to však nutné, takže klikneme na „*no thanks, just start my download*“.

Obrázek 16 - Webové stránky pro stažení MySQL

Tím je stažen soubor, který poté spustíme a začneme s instalací. Ráda bych upozornila na zadávání hesla k databázi. Je nezbytné si toto heslo pamatovat!

B.2 Připojení k serveru MySQL

Server je označení pro počítač, kde MySQL běží a kam si ukládá data. Existuje několik možností nastavení. Za prvé, klient je nastaven na Vašem počítači a server se nachází na jiném. Také se můžete z Vašeho počítače připojit na klienta MySQL, který se zase připojí k serveru MySQL. V této práci však předpokládám, že se klient i server nachází na stejném počítači.

Při prvním spuštění by server měl běžet - pokud si to při instalaci MySQL zvolíme. Může se však stát, že na tuto volbu zapomeneme.

Proto je třeba vědět, jak server spustit. Ve složce *Ovládací panely* najdeme položku *Nástroje pro správu* a v ní potom položku *Služby*. Pokud jste při instalaci nezvolili jinak, pak výchozí název serveru MySQL je MySQL56. Dvojklikem tedy otevřeme okno s vlastnostmi služby MySQL56 a spustíme server.

Další možností, jak spustit server, nabízí samotný program MySQL Workbench. Po přihlášení, ať už jako administrátor či jako běžný uživatel, se přepneme do záložky *Management*. Zvolíme možnost *Startup/Shutdown* a tím se otevře nová záložka. Nakonec klikneme na tlačítko *Start Server*.

Ne vždy je však tato volba k dispozici. Většinou musíme server spustit přes *Ovládací panely*.

B.3 Spuštění programu

Po instalaci program spustíme. Na úvodní obrazovce můžeme kromě jiného vidět například ukázku logického datového modelu (viz Obrázek 17). Také je tam seznam vytvořených připojení k databázi (*MySQL Connections*), který po instalaci obsahuje pouze jednu položku s názvem *Local instance MySQL56*. Toto připojení obsahuje práva administrátora.

B.4 Tvorba připojení, uživatele a databáze

Naším cílem je nejprve tvorba databáze. Abychom to mohli udělat, musíme se přihlásit jako administrátor, tedy spojit se s databází pomocí připojení *Local instance MySQL56*. Dvakrát poklikáme na dané připojení a systém nás vyzve k zadání hesla (viz Obrázek 18). Napíšeme heslo, které jsme zvolili při instalaci databáze.

Otevře se nová záložka (viz Obrázek 19). Nahoře vidíme dvě lišty. Jedna z nich nabízí širokou nabídku možností, které nástroj poskytuje. Tyto možnosti jsou podrobněji popsány v kapitole 3. Druhá lišta obsahuje ikony, které většinou slouží k tvorbě objektů (tabulky, pohledy, …) ale i databází neboli schémat.

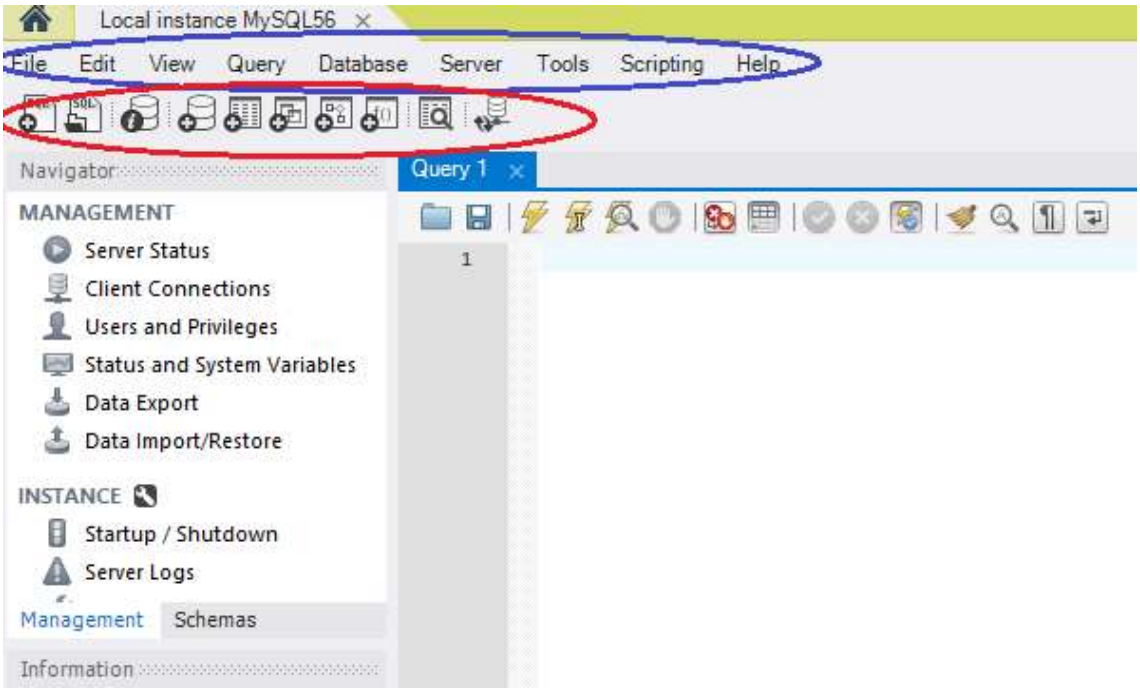
Po levé straně je tzv. *Navigator*. Ten obsahuje dvě záložky – *Management* a *Schemas*. Záložka *Schemas* je vlastně jakýmsi seznamem databází, ve kterých můžeme vytvářet tabulky, pohledy atd. Záložka *Management* slouží pro správu systému. Některé z nabízených možností budeme využívat, proto se k nim vrátím později. Pod *Navigatorem* je okno poskytující informace buď o vybraném objektu či o používaném připojení. Vedle okénka s informacemi je záložka s názvem *Query 1*. Těchto záložek může být otevřeno více, proto se číslují. Každá má však název *Query*. Prostor, který nabízí, je určen pro kód (příkazy) jazyka SQL. Pomocí příkazů jazyka SQL můžeme definovat objekty, manipulovat s daty, provádět transakce, atd. (Podrobněji viz kapitola 2.5). Pod záložkou *Query* je okno s názvem *Output*, kde můžeme sledovat historii zadaných příkazů či hlášení o úspěšnosti provedených akcí. V pravé části obrazovky je pak okno *SQL Additions*, které poskytuje jakousi nápovědu k syntaxi příkazů jazyka SQL.

V liště s ikonami (horní část obrazovky) klikneme na ikonu pro vytvoření nového schématu neboli databáze (viz Obrázek 20).

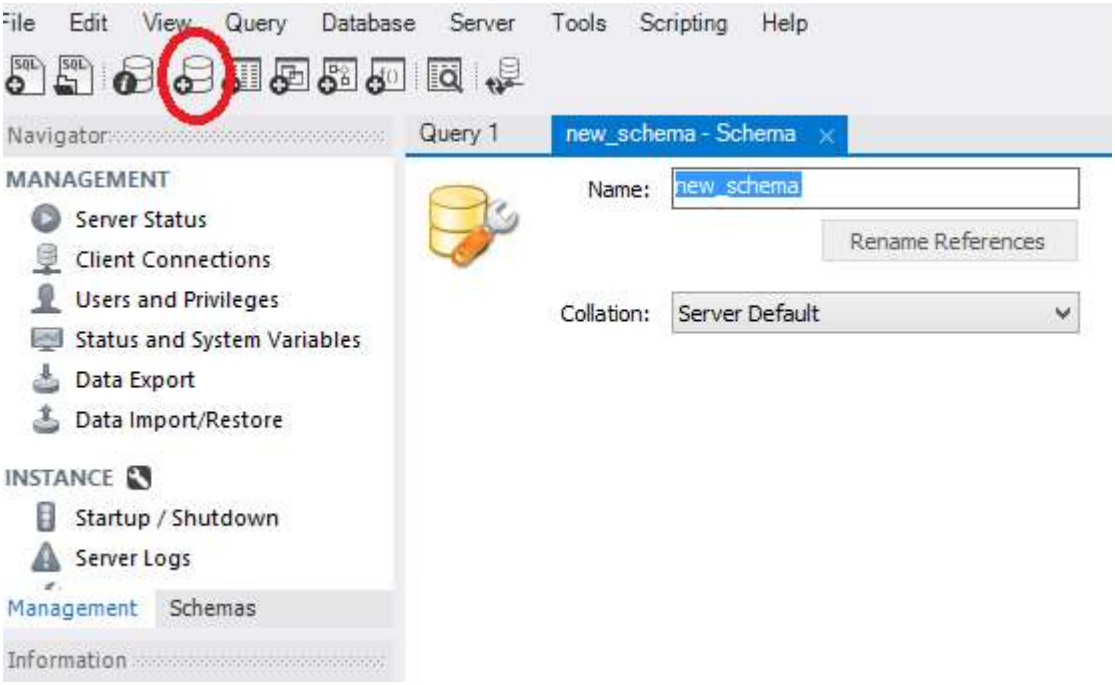
Otevře se záložka pro vytvoření nové databáze. Potvrdíme tlačítkem *Apply* a dokončíme kliknutím na tlačítko *Finish*. Záložku nyní můžeme zavřít.

Obrázek 17 - Úvodní obrazovka

Obrázek 18 - Zadání hesla pro připojení k databázi



Obrázek 19 - Obrazovka po připojení k databázi



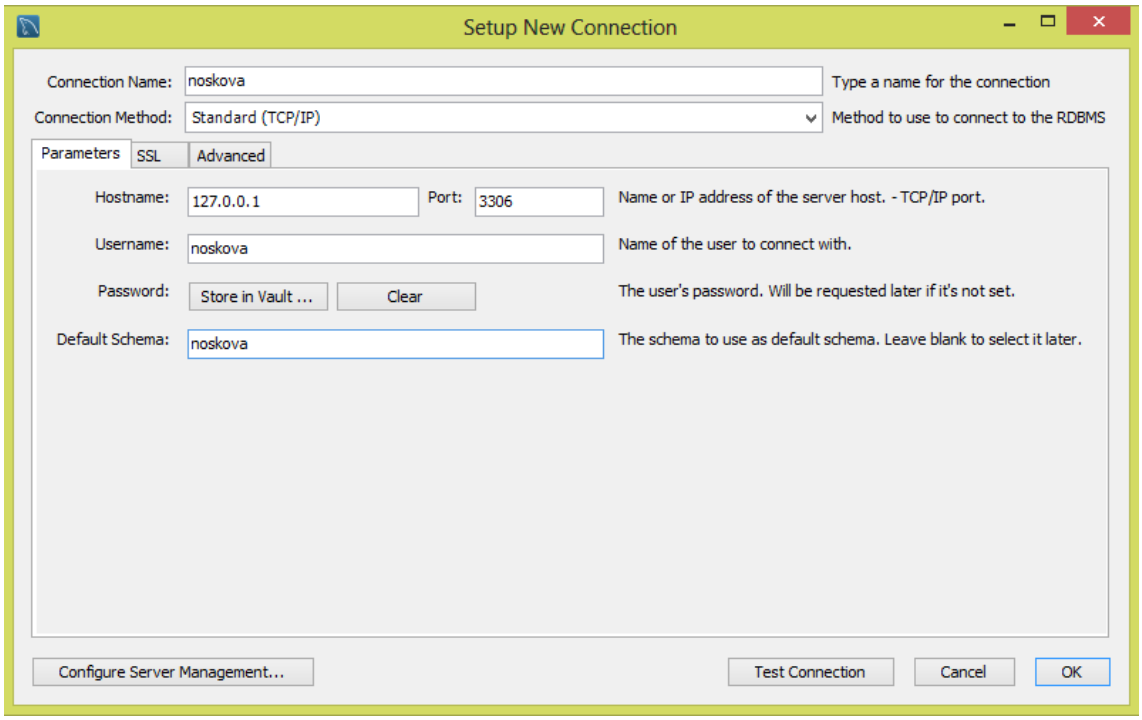
Obrázek 20 - Vytvoření databáze

Dále je třeba vytvořit účet pro běžného uživatele. V *Navigatoru* klikneme na *Users and Privileges*. Systém může opět vyžadovat heslo. V nově otevřené záložce (viz Obrázek 21) vidíme jednak seznam již vytvořených uživatelských účtů, ale také prostor pro definici nového účtu. Klikneme na tlačítko *Add Account* v dolní části obrazovky. V záložce *Login* definujeme login uživatele a heslo. V záložce *Schema Privileges* pak účtu přiřadíme databázi. Klikneme na *Add Entry…* a zvolíme výběr z již existujících schémat. Vybereme databázi, kterou jsme vytvořili, a klikneme na *OK*.

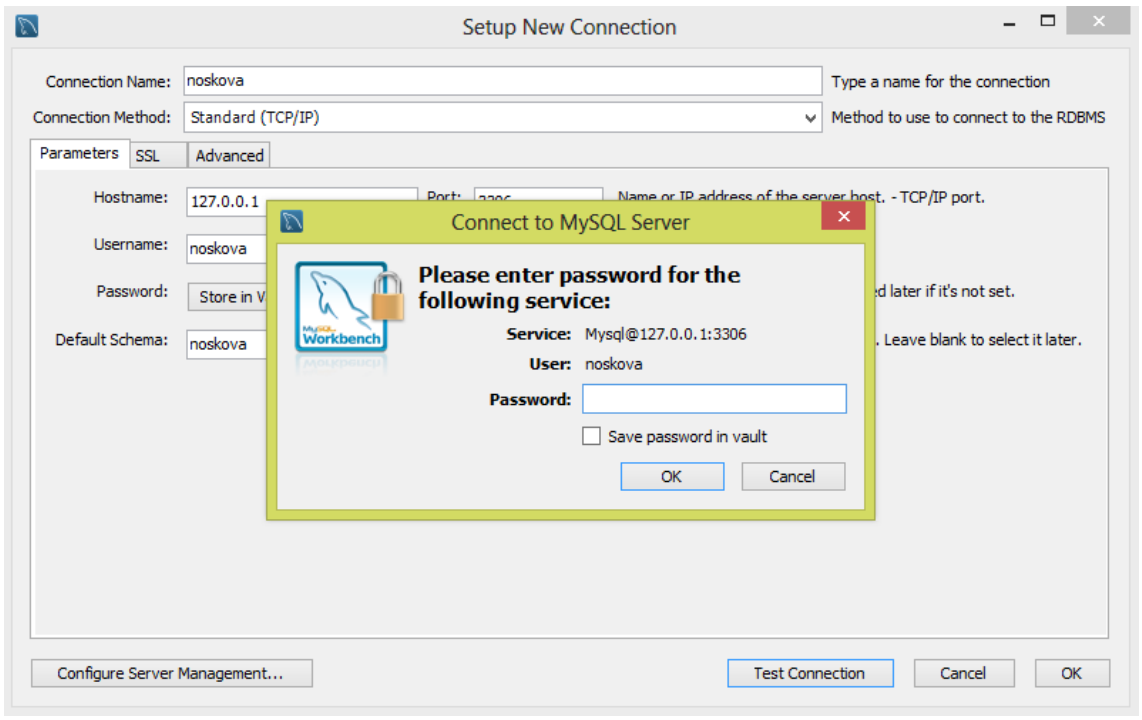
Posledním krokem je přiřazení práv. Rozhodneme, co bude smět uživatel v databázi dělat, jaké změny bude moci provádět, a ty zaškrtneme. Vše dokončíme kliknutím na tlačítko *Apply*. Nakonec je třeba vytvořit připojení k databázi pro běžného uživatele. To uděláme tak, že na úvodní obrazovce (viz Obrázek 17) klikneme na ikonu malého plus vedle textu *MySQL Connections*. Zobrazí se okno (viz Obrázek 22), kde budeme definovat nové připojení. Zvolíme název připojení, například login uživatele, a uživatelské jméno. Je dobré také zvolit výchozí databázi.

Klikneme na tlačítko *Test Connection* a systém nás vyzve k zadání hesla (viz Obrázek 23). Volba uložení hesla je libovolná. Nakonec klikneme na tlačítko *OK* a vytvořené připojení je přidáno do seznamu.

Obrázek 21 - Okno pro tvorbu uživatelského účtu



Obrázek 22 - Tvorba nového připojení



Obrázek 23 - Zadání hesla pro vytvoření připojení k databázi

Poté vytvoříme uživateli účet a přidělíme mu práva. Pro práci s databází je bezpečnější mít omezená práva.