VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

# NÁVRH DATABÁZE PRO ELTEX ELECTROCNIS

DATABASE DESIGN FOR ELTEX ELECTRONICS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — JITKA KRAVCOVÁ
AUTHOR

VEDOUCÍ PRÁCE — ING. JAN LUHAN, PH.D.
SUPERVISOR

BRNO 2014

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2013/2014
Ústav informatiky

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

**Kravcová Jitka**

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

**Návrh databáze pro Eltex electronics**

v anglickém jazyce:

**Database Design for Eltex Electronics**

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

BEGG, C., R. HOLOWCZAK, a T. CONOLLY. Mistrovství - Databáze: Profesionální průvodce tvorbou efektivních databází. Praha: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.
BRUCKNER, T., J. VOŘÍŠEK, A. BUCHALCEVOVÁ a kol. Tvorba informačních systémů: Principy, metodiky, architektury. Praha: Grada, 2012. 360 s. ISBN 978-80-247-4153-6.
GILMORE, W. J. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. 3. vyd. Brno: Zoner Press, 2011. 736 s. ISBN 978-80-7413-163-9.
CHURCHER, C. Beginning Database Design. 2nd ed. New York City: Apress, 2012. 252 p. ISBN 978-1-4302-4209-3.
SCHNEIDER, R. D.. MySQL Oficiální průvodce tvorbou, správou a laděním databází. 1. vyd. Praha: Grada, 2006. 372 s. ISBN 80-247-1516-3.
SCHWALBE, K. Řízení projektů v IT: Kompletní průvodce. Praha: Computer Press, 2011. 632 s. ISBN 978-80-251-2882-4.

Vedoucí bakalářské práce: Ing. Jan Luhan, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2013/2014.

L.S.

| | |
|---|---|
| doc. RNDr. Bedřich Půža, CSc.<br>Ředitel ústavu | doc. Ing. et Ing. Stanislav Škapa, Ph.D.<br>Děkan fakulty |

V Brně, dne 24.05.2014

# ABSTRAKT

Bakalářská práce se zabývá problematikou návrhu databázového systému přizpůsobenou požadavkům společnosti ELTEX electronics. Návrh databáze je realizován v databázovém systému MySQL. Databáze je navržena pro účely archivace a následné vizualizace dat získaných z měřicích přístrojů. Data jsou ukládána na databázový server, kde je připravený databázový systém MySQL s navrženou databází.



# ABSTRACT

Bachelor thesis deal with the proposal database system tailored to company requirements ELTEX electronics. Database design is realized in a database system MySQL. The database is designed for archiving and visualization of data obtained from measuring devices. Data is saved to the database server, where is the MySQL database system with the designed database.



# KLÍČOVÁ SLOVA

Databáze, MySQL, databázový systém, server, relační model, návrh databáze

# KEYWORDS

Database, MySQL, database system, server, relational model, database design

# BIBLIOGRAFICKÁ CITACE

KRAVCOVÁ, J. *Návrh databáze pro ELTEX electronics.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2014. 56 s. Vedoucí bakalářské práce Ing. Jan Luhan Ph.D.

# ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že předložená bakalářská práce je původní a zpracovala jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušila autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 14. května 2014

.....

Jitka Kravcová

# PODĚKOVÁNÍ

Ráda bych tímto poděkovala panu Ing. Janu Luhanovi Ph.D., který vedl mou bakalářskou práci, za ochotu, vstřícnost, věcné připomínky, čas a rady, které mi poskytnul během zpracovávání této bakalářské práce. Ráda bych také poděkovala společnosti ELTEX electronics za poskytnuté informace, podklady a konzultace při zpracování této bakalářské práce.

# OBSAH

ÚVOD ..... 11

CÍLE PRÁCE, METODY A POSTUPY ZPRACOVÁNÍ ..... 12

Vymezení problému práce ..... 12

Cíle práce ..... 12

Metody a postupy zpracování ..... 12

1 TEORETICKÁ VÝCHODISKA PRÁCE ..... 14

1.1 Základní pojmy ..... 14

1.2 Jazyk SQL ..... 15

1.3 MySQL ..... 16

1.3.1 Databázová úložiště MySQL ..... 16

1.4 Datové typy ..... 17

1.4.1 Číselné datové typy ..... 17

1.4.2 Řetězcové typy ..... 19

1.4.3 Typy pro datum a čas ..... 20

1.5 Relační model ..... 20

1.5.1 Model dat ..... 21

1.5.2 Terminologie ..... 21

1.5.3 Vlastnosti relačních tabulek ..... 22

1.5.4 Klíče relace ..... 22

1.5.5 Relační integrita ..... 23

1.6 Entitně-relační model ..... 23

1.6.1 Entita ..... 24

1.6.2 Atribut ..... 24

1.6.3 Relace (vztah) ..... 24

1.7 Normalizace ..... 25
1.7.1 První normální forma ..... 25
1.7.2 Druhá normální forma ..... 25
1.7.3 Třetí normální forma ..... 26
1.8 Metodologie návrhu ..... 26
1.8.1 Fáze návrhu databáze ..... 26
1.8.2 Zásady pro úspěšný návrh databáze ..... 27
1.8.3 Konceptuální návrh databáze ..... 27
1.8.4 Logický návrh databáze ..... 28
1.8.5 Fyzický návrh databáze ..... 28
1.9 Indexy ..... 29
1.10 Uložené procedury ..... 30
1.11 Zabezpečení databáze ..... 30
1.12 Provoz a údržba databáze ..... 31
1.13 Zálohování databáze ..... 31
2 ANALÝZA SOUČASNÉHO STAVU ..... 32
2.1 Požadavky společnosti ..... 32
2.1.1 Vizualizace dat ..... 32
2.1.2 Měřená data ..... 33
2.2 Síťová infrastruktura ..... 33
3 VLASTNÍ NÁVRHY ŘEŠENÍ ..... 35
3.1 MySQL Workbench ..... 35
3.1.1 Nová tabulka a její atributy ..... 35
3.2 Fáze konceptuálního návrhu ..... 36
3.2.1 Návrh entit ..... 36

3.2.2 Návrh relací ... 38
3.2.3 Shrnutí fáze konceptuálního návrhu ... 39
3.3 Fáze logického návrhu ... 39
3.3.1 Měřící stanice ... 40
3.3.2 Měřicí přístroj ... 41
3.3.3 Typ přístroje ... 41
3.3.4 Vstup ... 42
3.3.5 Jednotka ... 43
3.3.6 Rozsah ... 43
3.3.7 Hodnota ... 44
3.3.8 Čas ... 44
3.3.9 Aktuální hodnota ... 45
3.3.10 Pohled vstup ... 46
3.3.11 Pohled ... 46
3.3.12 Skupina pohledů ... 47
3.3.13 Nastavení zobrazení ... 48
3.3.14 Nastavení grafu ... 49
3.3.15 Shrnutí fáze logického návrhu ... 50
3.4 Fáze fyzického návrhu ... 50
4 ZÁVĚR ... 51
SEZNAM POUŽITÉ LITERATURY ... 52
SEZNAM TABULEK ... 54
SEZNAM OBRÁZKŮ ... 55
SEZNAM PŘÍLOH ... 56

# ÚVOD

Lidé měli vždy potřebu shromažďovat nejrůznější informace a data. K přehlednému shromažďování dat je vhodná databáze, ve které jsou tyto informace a data uchovávány systematicky a přehledně. Za databázi lze považovat i jednoduchý textový soubor s odřádkovanými informacemi, protože databáze je uspořádaná množina informací uložená na paměťovém médiu. Za součást databáze lze teoreticky považovat i software, díky kterému je možné k datům v databázi přistupovat či s nimi manipulovat. Odborně se tento software nazývá systém řízení báze dat.

Na databázových systémech je postavena moderní společnost ať chceme nebo ne. Informace o evidenci občanů, zdravotnictví, hospodářství, školství, letectví či výzkumu se shromažďují právě v databázích.

Databázových systémů je na trhu několik od různých značek. Mezi databázové systémy na trhu patří například: Oracle, DB2, Microsoft Access, Microsoft SQL server, MySQL a další.

# CÍLE PRÁCE, METODY A POSTUPY ZPRACOVÁNÍ

## Vymezení problému práce

Společnost ELTEX electronics potřebuje navrhnout databázi pro účely archivace a následné vizualizace dat získaných z nejrůznějších měřicích přístrojů. Data jsou získávána z měřicích přístrojů pomocí aplikací na několika měřících počítačových stanicích. Všechny tyto měřící počítačové stanice jsou společně s databázovým serverem umístěny v jedné síťové infrastruktuře. Na databázovém serveru je připraven databázový systém MySQL, proto bude návrh databáze zpracováván v MySQL. Pro snadnější zpracovávání návrhu databáze jsem si vybrala MySQL Workbench, protože tento software umožňuje vizuální zpracování návrhu, což je přehledné a je jednodušší se v takovémto návrhu zorientovat.

## Cíle práce

Cílem předkládané práce je návrh databázového řešení přizpůsobeného potřebám a požadavkům společnosti ELTEX electronics. Databáze bude realizována v databázovém systému MySQL, který má firma k dispozici na databázovém serveru. Databáze bude sloužit pro archivaci a vizualizaci dat získaných z měřicích přístrojů. Databáze bude umístěna na databázový server v síťové infrastruktuře, na kterou jsou připojené i měřící počítačové stanice, které komunikují s měřicími přístroji. Databáze bude navrhnuta tak, aby data byla ukládána i v případě výpadku serveru nebo přerušení komunikace měřicích přístrojů s databázovým serverem. Databáze musí být navrhnuta tak, aby po obnovení komunikace mezi měřícími stanicemi a serverem byla možnost synchronizace dat z měřících stanic na databázový server.

## Metody a postupy zpracování

Při zpracování návrhu databáze bude vycházeno z požadavků společnosti ELTEX electronics. Pro návrh databáze bude využito konceptuální metodologie a ER diagramu. Konceptuální metodologie má tři základní fáze: konceptuální, logickou a fyzickou fázi.

Návrh databáze bude zpracováván v programu MySQL Workbench verze 6.0, protože umožňuje vizuální zpracovávání návrhu databáze. V takovémto návrhu se snadněji

orientuje a mezi jednotlivými navrženými entitami jsou ihned viditelné vztahy mezi nimi. Mezi výhody programu MySQL Workbench patří i možnost automatického vygenerování skriptu pro vytvoření navržené databáze.

# 1 TEORETICKÁ VÝCHODISKA PRÁCE

Tato část předkládané práce bude zaměřena na teoretická východiska týkající se databází, jako je definování pojmů: databázový systém, databáze, systém řízení databáze, databázové aplikace. Dále bude práce v této části zaměřena na samotné MySQL, ve kterém bude zpracovávána praktická část předkládané práce.

## 1.1 Základní pojmy

Databázový systém je možné popsat jako soubor databáze, systému řízení databáze a kolekce databázových aplikací interagujících s databází [9].

Databázi lze definovat jako velké úložiště dat, která mohou být používána současně více uživateli. Data, se kterými uživatelé pracují a která požadují, jsou integrována s minimálním množstvím duplikací. Databáze není vlastněna žádným uživatelem, ale je sdíleným zdrojem dat společnosti [9]. Základem databáze je tabulka, která má sloupce a řádky. Sloupce tabulky představují vlastnosti jednotlivé položky a řádky představují vlastnosti příslušející jednotlivým položkám [4].

Je důležité uvědomit si rozdíl mezi pojmy „data“ a „informace“. Data jsou surová fakta mající určitou důležitost pro jednotlivce či organizaci, ale nemají význam. Informace jsou data, která prošla zpracováním či dostala strukturu, která jim dává význam. Toto rozlišení pojmů data a informace není univerzální a často dochází k jejich záměně [9].

Databáze neobsahuje jen provozní data, ale obsahuje i popis těchto provozních dat, proto je databáze definována i jako sebe popisující kolekce integrovaných záznamů. Tato data o datech se nazývají metadata, dále se označují jako slovník dat či systémový katalog [9].

Při analýze informačních potřeb organizace, identifikujeme důležité objekty a logické vztahy mezi těmito objekty, které je třeba reprezentovat v databázi. Na identifikaci těchto důležitých objektů a logických vazeb mezi nimi existuje několik metodologií [9].

Dalším důležitým pojmem je systém řízení databáze. Jedná se o softwarový systém umožňující uživateli definovat, vytvářet a udržovat databázi a poskytuje řízený přístup k této databázi. Tento software interaguje s uživateli, databázovými aplikacemi a se samotnou databází. Systém řízení databáze umožňuje uživatelům vkládat, aktualizovat, mazat ale i volat data z databáze. Existuje-li centrální skladiště dat a popis

dat, pak může systém řízení databáze poskytnout všeobecnou možnost dotazování na tato data pomocí dotazovacího jazyku (jako je například SQL). Systém řízení databáze lze rozdělit na pět základních složek: hardware (počítačový systém/systémy na kterém běží databáze), software, data (slouží jako můstek mezi hardwarovou a softwarovou složkou a lidskou složkou), procedury (instrukce a pravidla řídící návrh a používání databáze) a osoby (analytici, návrháři, správci, programátoři a uživatelé) [9].

Databázová aplikace je počítačový program interagující s databází vyvoláním odpovídajícího požadavku jednoho či více SQL příkazů. Uživatelé pracují s databází pomocí několika databázových aplikací – pro vytváření a správu databáze, ale i pro generování informací. Tyto aplikace mohou být dávkovými aplikacemi nebo online aplikacemi. Databázové aplikace lze programovat v jazycích třetí či vyšší generace. Mezi jazyky třetí generace patří například C++ nebo Java [9].

Pohled je virtuální tabulka, kterou generuje systém řízení databáze z tabulek obsažených v databázi. Pohled je definován jako dotaz zpracovávající základní tabulky, aby vytvořil virtuální tabulku, která redukuje složitost například tím, že uživatel vidí jen vybraná data, která vidět chce či potřebuje. Mezi další výhody pohledů patří poskytování zabezpečení (uživatel nevidí všechna data), poskytování mechanismu pro přizpůsobení vzhledu databáze, vytvoření konzistentního a neměnného obrazu struktury databáze (i přes změny základní databáze) [9].

## 1.2 Jazyk SQL

SQL je anglická zkratka pro Structured Query Language, což je programovací jazyk [4]. Historie SQL jazyka sahá až do sedmdesátých a osmdesátých let. První standard byl přijat v roce 1986, časem se však projevily nedostatky. Opravná verze byla vydána až v roce 1992 a stala se standardem relačních databází až do dnes. Jazyk SQL zahrnuje nástroje pro tvorbu tabulek a vztahů mezi nimi, ale i nástroje pro manipulaci s daty (vkládání, aktualizace či mazání dat, nebo vyhledávání informací) [11]. Jazykem SQL se pomocí klienta ptáme na data SQL serveru, který následně při správně položeném dotazu odpoví. Odpověď je formulována pomocí tabulky o jednom čí více řádcích. Může nastat i situace, kdy odpovědní tabulka nebude mít žádné řádky, a to v případě, že ani jedna položka z databáze nevyhovuje všem podmínkám dotazu zároveň [4].

Databáze standardu SQL jsou založeny na komunikaci server-klient. Jako server vystupuje počítač, na němž je spuštěna služba naslouchající na portu určeném konfigurací. Služba je připravena naslouchat a dávat odpovědi na správně položené dotazy. Klient je počítač s aplikací, který dotazy pokládá [4].

## 1.3 MySQL

MySQL je relační databázový systém šířený open source pro nekomerční použití [1]. Jedná se o vysoce výkonný relační databázový model, který umí uchovávat velké množství dat bez velké ztráty výkonu. MySQL lze využívat téměř na všech dnes používaných platformách jako je Windows, Linux, OS X a další [6].

MySQL implementuje spoustu funkcí s dostatečnou rychlostí, ale do verze 5.0 nepodporoval transakce. Transakce je jeden čí více SQL příkazů provedených jedním uživatelem, které končí buď úspěšnou změnou v databázi, nebo uvedením databáze zpět do výchozího stavu, přičemž pro ostatní uživatele databáze musí být tato změna patrná až po úspěšném vykonání celého sledu instrukcí [4].

### 1.3.1 Databázová úložiště MySQL

Databázová úložiště tvoří srdce databázového systému MySQL, protože jsou zodpovědná za ukládání a načítání informací do databáze. MySQL nabízí spoustu možností, pokud jde o výběr specializovaného úložiště nebo druhu tabulky. Mezi druhy úložišť patří především MyISAM, InnoDB, MERGE, MEMORY, ARCHIVE, CSV, NDB Cluster, ISAM a BDB [3].

**MyISAM** je rychlé, komprimovatelné implicitní úložiště MySQL s fulltextovým vyhledáváním. **InnoDB** je robustní úložiště umožňující transakční zpracování se silnou referenční integritou, které je často používáno pro složité aplikace s velkým objemem dat, v nichž je nezbytné použití transakcí. Vytvořením jediného pohledu na více identických tabulek MyISAM je **MERGE**, toto úložiště je základem pro hlášení o zaplnění nebo pro nástroje Decision Support System (DSS) či Online Analytical Processing (OLAP). Datové úložiště typu **MEMORY** je extrémně rychlé, jednoduše konfigurovatelné a umožňuje vývojářům využít výhod práce s pamětí, kde jsou tabulky tohoto typu uložené. Úložiště **ARCHIVE** je zaměřeno na aplikace s velmi velkým objemem nepříliš často aktualizovaných informací. Tabulky úložiště ARCHIVE šetří místo na disku. Vytvořením

souborů s daty oddělenými čárkami usnadňuje úložiště **CSV** vývojářům plnění jiných aplikací, které umí pracovat s tímto druhem souborů s daty MySQL. **NDB Cluster** je základní úložiště pro MySQL Cluster, které umožňuje uchovávat synchronizovaná data na více počítačích, což vede k obrovské škálovatelnosti a zvýšení výkonu. **ISAM** je původní úložiště MySQL, které bylo postupně nahrazeno úložištěm MyISAM. **BDB** bylo prvním úložištěm MySQL, které podporovalo i transakce. Úložiště InnoDB však získalo větší podíl na trhu. Nejpopulárnějšími úložišti jsou MyISAM a InnoDB [3].

## 1.4 Datové typy

Datové typy lze rozdělit do tří základních skupin: číselné datové typy, řetězcové datové typy a datové typy pro datum a čas. Tyto tři základní skupiny datových typů zahrnují mnoho datových typů. Práce se bude zaměřovat na ty nejznámější a nejdůležitější. Celkem bude v této práci popsáno devatenáct datových typů. Tabulky datových typů jsou pro lepší přehlednost souhrně i v příloze práce

### 1.4.1 Číselné datové typy

MySQL podporuje několik datových typů pro čísla, které je možné rozdělit do dvou podskupin. První podskupinou datových typů My SQL jsou datové typy pro celá čísla (celočíselné datové typy), druhá podskupina datových typů je pro desetinná čísla (číselné datové typy s pohyblivou desetinnou čárkou) [1].

#### 1.4.1.1 Celočíselné datové typy

Mezi celočíselné typy patří: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT. U celočíselných datových typů lze nastavit maximální zobrazovací šířku a není-li uvedeno jinak, jedná se o celé číslo od 1 do 255 [7].

**TINYINT** se používá se pro velmi malá celá čísla. **SMALLINT** se používá pro malá celá čísla. **MEDIUMINT** je určen pro středně velká celá čísla. **INT** se používá pro běžně velká celá čísla. **BIGINT** se využívá pro velká celá čísla v rozsahu −9223372036854775808, až 9223372036854 uvažujeme-li záporná čísla, nebo v rozsahu 0 až 18446744073709551615 uvažujeme-li pouze kladná čísla. V paměti zabírá pro každé číslo osm bajtů [7].

Tabulka 1: Celočíselné typy (Zdroj [1])

| Datový typ | Rozsah<br>se znaménkem | bez znaménka | Velikost | Výchozí hodnota |
|---|---|---|---|---|
| **TYNYINT** | -128 až 127 | 0 až 255 | 1 bajt | Null / 0 |
| **SMALLINT** | -32768 až 32767 | 0 až 65535 | 2 bajty | Null / 0 |
| **MEDIUMINT** | -8388608 až 8388607 | 0 až 16777215 | 3 bajty | Null / 0 |
| **INT** | -2147483648 až 2147483647 | 0 až 4294967295 | 4bajty | Null / 0 |

#### 1.4.1.2 Datové typy s pohyblivou desetinnou čárkou

Typy s pohyblivou desetinou čárkou: FLOAT, DOUBLE, DECIMAL. U těchto typů lze nastavovat maximální zobrazovací šířku jako u celočíselných datových typů, navíc lze ještě nastavit počet číslic za desetinnou čárkou. Počet číslic za desetinnou čárku nesmí být větší než maximální zobrazovací šířka mínus dva, jinak se automaticky upraví maximální zobrazovací šířka na počet desetinných míst plus dva. Je-li počet číslic za desetinnou čárkou nastaven na rovno nule, pak data nemají desetinnou čárku, ani desetinnou část čísla. Je-li použit parametr UNSIGNED, nejsou povolené záporné číselné hodnoty [7].

Tabulka 2: Číselné typy s pohyblivou desetinnou čárkou (Zdroj [1])

| Datový typ | Rozsah<br>se znaménkem | Velikost | Výchozí hodnota |
|---|---|---|---|
| **FLOAT** | ±1,175494351E−38 až ±3,402823466E+38 | 4 bajty | Null / 0 |
| **DOUBLE** | ±2,2250738585072014E-308 až ±1,17976931348623157E+30 | 8 bajtů | Null / 0 |
| **DECIMAL** | ±2,2250738585072014E-308 až ±1,17976931348623157E+30 | pohyblivé | Null / 0 |

**FLOAT** se používá pro malá čísla v pohyblivé řádové čárce. Není-li určena maximální zobrazovací šířka ani počet číslic za desetinnou čárkou, pak není nastavena velikost

zobrazení ani přesnost. **DOUBLE** se využívá pro ukládání velkých čísel s pohyblivou řádovou čárkou. Je přesnější než datový typ FLOAT. Není-li určena maximální zobrazovací šířka ani počet číslic za desetinnou čárkou, pak není nastavena velikost zobrazení ani přesnost stejně jako u datového typu FLOAT. **DECIMAL** je používán pro velká čísla v pohyblivé řádové čárce ukládané jako řetězec. Pokud není nastaveno jinak, pak je výchozí hodnotou pro maximální zobrazovací šířku deset a výchozí hodnota pro počet číslic za desetinnou čárkou nula. V paměti zabírá pohyblivý počet bajtů. Jeden bajt pro každou číslici, jeden bajt pro desetinnou čárku je-li použita a jeden bajt pro znaménko mínus je-li použito [7].

### 1.4.2 Řetězcové typy

Pro zpracování řetězců má MySQL několik datových typů: CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT [1]. Používají se pro ukládání textu. Lze do nich ale ukládat i čísla a další data. Jednotlivé typy mají omezenou maximální délku a některé rozlišují i velikost písmen. U typu CHAR a VARCHAR lze nastavit maximální zobrazovací šířku (není-li nastaveno jinak, jedná se o celé číslo od 1 do 255). U všech je výchozí hodnota NULL nebo ‘’ (tj. prázdný řetězec). Jen typ VARCHAR rozlišuje s parametrem BINARY velikost písmen uložených v řetězci [7].

Tabulka 3: Řetězcové typy (Zdroj [7])

| Datový typ | Povolená délka | Velikost |
|---|---|---|
| **CHAR** | 0 až 255 bajtů | až 255 bajtů |
| **VARCHAR** | 0 až 255 bajtů | hodnota délky + 1 bajt |
| **TINYTEXT** | 0 až 255 bajtů | hodnota délky + 1 bajt |
| **TEXT** | 0 až 65535 bajtů | hodnota délky + 2 bajty |
| **MEDIUMTEXT** | 0 až 16777245 bajtů | hodnota délky + 3 bajty |
| **LONGTEXT** | 0 až 4294967295 bajtů | hodnota délky + 4 bajty |

Pokud u typu **CHAR** není nastavena maximální zobrazovací šířka, pak se automaticky nastaví tato hodnota na jedna. Je-li ukládaný řetězec delší než nastavená maximální zobrazovací šířka, pak se tento ukládaný řetězec zkrácen a zbytek řetězce je ignorován.

Je-li ukládaný řetězec kratší než maximální zobrazovací šířka, pak je tento řetězec doplněn mezerami na maximální zobrazovací šířku. Při načítání dat z databáze jsou tyto nadbytečné mezery ale ignorovány. U typu **VARCHAR** je dlouhý ukládaný řetězec přesahující nastavenou maximální zobrazovací šířku zkrácen a zbytek je ignorován. Je-li ukládaný řetězec kratší než maximální zobrazovací šířka, koncové mezery se před uložením odstraní [7].

### 1.4.3 Typy pro datum a čas

Datum a čas lze v databázi také uchovávat v několika typech: DATE, DATETIME, TIMESTAMP, TIME, YEAR [1].

Tabulka 4:Typy pro datum a čas (Zdroj [1])

| Datový typ | Formát | Rozsah | Velikost |
|---|---|---|---|
| **DATE** | 'RRRR-MM-DD' | '1000-01-01' až '9999-12-31' | 3 bajty |
| **DATETIME** | 'RRRR-MM-DD hh:mm:ss' | '1000-01-01 00:00:00' až '9999-12-31 23:59:59' | 8 bajtů |
| **TIMESTAMP** | 'RRRR-MM-DD hh:mm:ss' | '1970-01-01 00:00:00' až '2037-01-01 00:00:00' | 4 bajty |
| **TIME** | 'hh:mm:ss' a '-hh:mm:ss' | '-838:59:59' až '838:59:59' | 3 bajty |
| **YEAR (4)** | RRRR | '1901' až '2155' a '0000' | 1 bajt |
| **YEAR (2)** | RR | '1970' až '2069' | 1 bajt |

Pro datový typ **YEAR** je výchozí maximální zobrazovací šířka čtyři, je možné ale nastavit zobrazování pouze posledních dvou číslic. **TIMESTAMP** udržuje vždy aktuální čas poslední změny záznamu [7].

## 1.5 Relační model

V relačním modelu jsou data logicky strukturovaná do relací (tabulek). Relace je jednoduchá a logická struktura, která je silnou stránkou relačního modelu. Relační model navrhl E. F. Codd v roce 1970. Cíle relačního modelu byly umožnit vysoký stupeň

nezávislosti dat, poskytnout základ pro normalizaci relací a umožnit expanzi množinově orientovaných jazyků pro manipulaci s daty [9].

Komerční systémy založené na relačním modelu se začaly objevovat až koncem sedmdesátých a začátkem osmdesátých let dvacátého století. V současné době existuje několik set relačních systémů řízení báze dat a mnoho z nich nedodržuje striktně definici relačního modelu. Vznikly i návrhy některých rozšíření relačního modelu, aby byl lépe vystihnut význam dat, aby se podpořily objektově orientované koncepty, aby byly lépe podporovány deduktivní schopnosti [9].

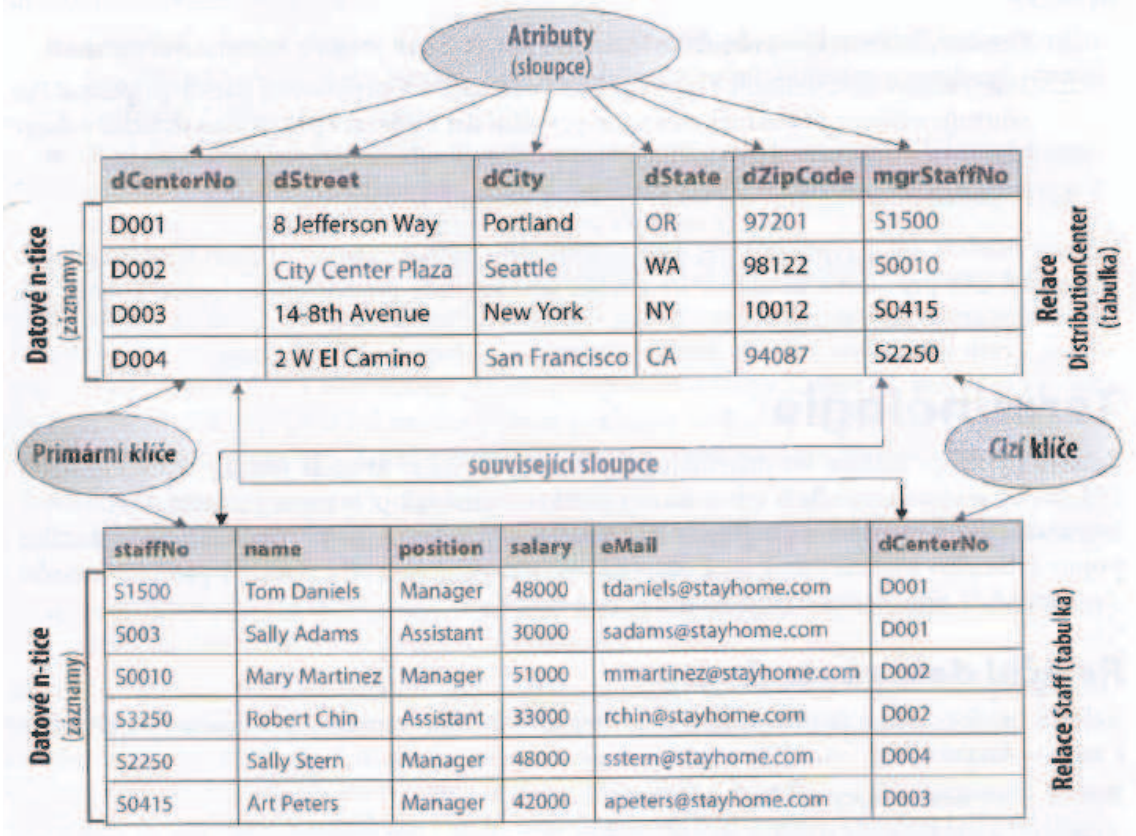
### 1.5.1 Model dat

Model dat je integrovaná kolekce konceptů pro popis dat, relací mezi nimi a jejich omezení. Jedná se o data používaná organizací. Model dat je reprezentace reálného světa objektů a událostí a jejich souvislostí. Nahodilé vlastnosti jsou ignorovány a naopak podstatné a základní aspekty organizace jsou konkretizovány. Úkolem modelu dat je reprezentace datových požadavků organizace nebo její části. Měl by poskytovat základní koncepty a charakteristiky, které umožní návrhářům databáze a koncovým uživatelům jednoznačně a přesně komunikovat o organizaci dat. Datový model má tři části: strukturální část (obsahuje množinu pravidel jak konstruovat databázi), manipulační část (definující typy operací přípustných na datech) a množinu pravidel integrity (zajišťující přesnost dat). Účelem je data reprezentovat srozumitelným způsobem [9].

### 1.5.2 Terminologie

Relační model je založen na matematickém konceptu relace, kterou fyzicky reprezentuje tabulka. Codd byl matematikem a tak využil matematickou terminologii (včetně teorie množin a predikátové logiky [9].

Relační model má pět základních složek: relaci, atribut, datovou n-tici, doménu a relační databázi. Relaci představuje tabulka, kterou využíváme k ukládání informací o objektech, které chceme v relaci reprezentovat. Řádky této tabulky odpovídají datovým n-ticím a sloupce atributům. Na pořadí atributů v relaci nezáleží, i když jsou jinak uspořádané, jedná se o stejnou relaci. Domény představují množinu přípustných hodnot pro atributy a definuje tak jejich význam a zdrojové hodnoty. S jednou doménou může být spojeno několik atributů. Relační databáze se skládá z tabulek s odpovídající strukturou (tabulky jsou normalizované) [9].




Obrázek 1: relační datová struktura (Zdroj [9])

### 1.5.3 Vlastnosti relačních tabulek

Relační tabulka má následující vlastnosti: jméno, které ji odlišuje od všech ostatních tabulek v dané databázi, každá buňka tabulky obsahuje přesně jednu hodnotu, každý atribut má jedinečné jméno, všechny hodnoty v jednom atributu (sloupci) jsou ze stejné domény, pořadí atributů (sloupců) v tabulce nemá význam, neexistují duplicitní záznamy v tabulce - každý záznam je jedinečný [9].

### 1.5.4 Klíče relace

Za klíč relace je považován atribut či kombinace atributů z jedné tabulky zajišťující jedinečnost záznamu. Rozlišujeme především kandidátní klíč, primární klíč a cizí klíč. Kandidátní klíč je takový klíč, který obsahuje minimální počet atributů nutných k jedinečné identifikaci záznamů. Kandidátní klíč se tedy vyznačuje dvěma vlastnostmi – jedinečnost a neredukovatelnost [9].

Primární klíč je vybrán z kandidátních klíčů. Protože tabulka nemá duplicitní záznamy je díky primárnímu klíči velice snadné najít kterýkoliv konkrétní záznam. Každá tabulka má svůj primární klíč [9].

Cizí klíč je sloupec či skupina sloupců v jedné tabulce, která odpovídá kandidátnímu klíči jiné tabulky. Cizí klíč v jedné tabulce tvoří s primárním klíčem v jiné tabulce vztah mezi těmito tabulkami (cizí klíč v jedné tabulce je shodný s primárním klíčem v tabulce jiné) [9].

### 1.5.5 Relační integrita

Protože každý atribut tabulky je spojen doménou, existují omezení množiny přípustných hodnot pro daný atribut tabulky. Navíc existují ještě další dvě důležitá integritní omezení, která se vztahují na všechny instance databáze. Integritní omezení jsou pravidla definující nebo omezující některé vlastnosti dat užívaných organizací. Tato pravidla omezují například domény – jejich hodnoty.

Jedná se o entitní integritu a referenční integritu. Nejprve si ale objasníme pojem hodnoty null [9].

Hodnoty null představují neznámé nebo neplatné hodnoty pro daný záznam. Může znamenat, že pro daný atribut daná hodnota nelze použít, nebo že hodnota nebyla zadána. „Null" není považována za hodnotu ale za nepřítomnost hodnoty, na rozdíl od číselné nuly nebo mezery (to jsou hodnoty) [9].

První integritní pravidlo je entitní integrita – vztahuje se na primární klíče tabulek (fyzicky uložených v databázi, ne pohledových tabulek). Toto pravidlo říká, že v tabulce atribut primárního klíče nesmí mít prázdnou hodnotu. Druhým integritním pravidlem je referenční integrita – existuje-li v tabulce cizí klíč, pak musí jeho hodnota odpovídat hodnotě některého z primárních klíčů jiné tabulky nebo musí mít cizí klíč prázdnou hodnotu [9].

## 1.6 Entitně-relační model

Základními elementy ER modelu jsou na konceptuální úrovni entity a vztahy (relace). Za entitu je považována věc, která může být samostatně identifikována (například konkrétní společnost, osoba, událost atd.). Vztah je nějaké spojení mezi entitami. U vztahů lze rozlišovat jejich kardinalitu a volitelnost. Kardinalita je vlastnost vztahu, lze rozlišovat, zda vztah jedné entity k druhé je výjimečný a může nastat tedy pouze jednou, nebo zda může tento vztah nastat vícekrát. U volitelnosti vztahu sledujeme povinnost (zda vztah nastává vždy) či volitelnost (vztah nastat nemusí) vztahu [10].

### 1.6.1 Entita

Entita je rozlišitelný a identifikovatelný objekt světa, který popisujeme. Entity lze na základě podobnosti slučovat do entitních množin. Každá entitní množina má uveden identifikátor a má uvedenou i minimální množinu atributů, které zajišťují jednoznačnou identifikaci entity v této množině. Nejčastěji se rozlišují tyto typy entit: obecná, základní (nezávislá na jiných entitách), popisná (existenčně závislá na jedné či více jiných entitách, přičemž existenční závislosti jsou vyjádřeny pomocí charakteristik vztahů), vazební (popisuje vazbu mezi entitami), generalizační (vytváří entitu vyšší úrovně z dvou či více entit nižší úrovně), specializační (vytváří jednu či více entit nižší úrovně z jedné entity úrovně vyšší) [10].

### 1.6.2 Atribut

Atribut je datový prvek, který blíže charakterizuje entitu nebo vztah mezi entitami. I atributy se popisují na typové úrovni (stejně jako entity). Atributy lze dělit podle povinnosti: povinný (totální) atribut – pro každý výskyt entity má tento atribut hodnotu, nebo je atribut nepovinný (parciální) atribut – atribut nemá hodnotu pro každý výskyt entity. Dále lze dělit atributy dle druhů: základní atribut – nelze jej odvodit z jiných atributů, odvoditelný atribut – je možné získat jej odvozením a jiných základních či odvoditelných atributů. Tyto odvoditelné atributy se do datových modelů většinou nezakreslují [10].

### 1.6.3 Relace (vztah)

Jednotlivé entity mezi sebou mají vzájemné konkrétní vztahy. Pro tuto práci se zaměříme pouze na vztahy kardinální – tzn. maximální a minimální počet výskytů entity v určitém vztahu. Rozlišujeme vztahy 1:1, 1:N a M:N [10].

Vztah 1:1 přiřazuje záznamu z jedné tabulky právě jeden záznam z tabulky jiné. Vztah 1:N umožňuje jednomu záznamu z jedné tabulky přiřadit více záznamů z jiné tabulky. Vztah M:N není moc častý, umožňuje několika záznamům z jedné tabulky přiřadit několik záznamů z jiné tabulky. Při vztahu M:N se provádí dekompozice pomocí vazebné tabulky – vzniknou tak místo jednoho vztahu M:N vztahy dva 1:N [9].

## 1.7 Normalizace

Pro návrh datového modelu databáze existuje celá řada pravidel. Nejdůležitější je normalizace dat, která má předcházet ukládání duplicitních informací do databáze. Další pravidla se týkají například referenční integrity a výběru indexů. Některá pravidla pro návrh datového modelu databáze jsou však protichůdná, proto nemá smysl dodržovat striktně všechna pravidla a občas je potřeba volit kompromis [5].

Normalizací jako techniku vyvinul E. F. Codd v roce 1972. Používá se pro vytvoření sady tabulek databáze s minimální redundancí. Často se normalizace provádí jako soubor testů na tabulce, aby se určilo, zda jsou dodržena pravidla pro určitou normální formu. Normalizačních forem je několik, nejpoužívanější z nich jsou: první normální forma, druhá normální forma a třetí normální forma. Všechny tyto formy jsou založeny na vztazích mezi sloupci tabulky [9].

Normalizační formy vychází z požadavků na efektivní ukládání dat a minimalizují redundance při zachování integrity a konsistence dat. Datový model porušující některou z normalizačních forem není navržen optimálně. Je-li databáze normalizována na vyšší úrovni, pak musí být normalizována na všech předešlých. Normalizace se provádí tak, aby byly zachovány následující požadavky: bezztrátovost dat při zpětném spojení, závislosti, redundance (odstranění duplicitních záznamů) [11].

### 1.7.1 První normální forma

Tabulka je v první normální formě, pokud jsou všechny její atributy definovány nad doménami. To znamená, že všechny atributy entity jsou jednoduché (nikoliv složené či vícehodnotové) [11]. První normální forma je jedinou normalizační formou, která je nezbytná pro vytvoření vhodných tabulek pro relační databázi. Ostatní normalizační formy jsou volitelné. Obvykle se ale dodržují nejpoužívanější tři normální formy [9].

### 1.7.2 Druhá normální forma

Druhá normální forma se týká pouze tabulek se složenými primárními klíči (to znamená, že primární klíč je složen ze dvou či více atributů) [9]. Tabulka splňuje druhou normální formu, když je v první normální formě a když jsou všechny její atributy závislé na celém primárním (kandidátním) klíči [11].

### 1.7.3 Třetí normální forma

Tabulka je ve třetí normální formě, pokud splňuje první a druhou normální formu a pokud navíc nemá tranzitivní závislosti. To znamení, že tabulka má všechny neklíčové atributy vzájemně nezávislé [11]. Tranzitivní závislost (zprostředkovaná závislost) je taková závislost, kdy sloupec „a“ determinuje sloupec „b“ a sloupec „b“ determinuje sloupec „c“. Sloupec „c“ je tedy zprostředkovaně závislý na sloupci „a“ díky sloupci „b“ [9].

## 1.8 Metodologie návrhu

Metodologie návrhu je strukturovaný přístup používající procedury, techniky, nástroje a dokumentaci s cílem podpořit a usnadnit proces návrhu databáze. Metodologie se skládá z několika stádií, stádia se skládají z kroků a kroky vedou návrháře databáze odpovídajícími technikami ke každému stádiu. Jednotlivá stádia napomáhají návrháři při plánování, správě, kontrole a zhodnocení projektu. Jedná se o strukturovaný přístup k analýze a modelování souboru požadavků standardizovaným a organizovaným způsobem [9].

Návrh databáze je iterativní proces, který má počáteční bod a nekonečnou množinu možností zjemňování. Jedná se sice o procedurální postup, ale není nutné tento postup zcela striktně dodržovat. Je pravděpodobné, že některé znalosti získané v určitém kroku povedou ke změně rozhodnutí v krocích předcházejících. Metodologie tedy může sloužit jako rámec, který nás má efektivně vést procesem návrhu databáze [9].

Pro tuto práci jsem si vybrala konceptuální metodologii, která bude podrobněji popsána níže.

### 1.8.1 Fáze návrhu databáze

Konceptuální metodologie návrhu databáze rozděluje proces navržení databáze do tří hlavních fází: konceptuální, logickou a fyzickou. Během fáze konceptuálního návrhu se vytváří konceptuální model na základě dat používaných organizací bez jakéhokoli uvažování o podrobnostech nebo o implementaci. Tento model má za úkol identifikovat důležité entity a relace, které je potřeba v databázi reprezentovat. Konceptuální fáze slouží jako zdroj informací pro fázi logickou [9].

Logická fáze je proces vytvoření modelu dat využívaných organizací – je založen na specifickém modelu dat, ale jinak je nezávislý (na systému řízení báze dat i na implementaci). Vytváří se logická reprezentace databáze, jako základ je využíván v této práci relační model dat (popsán dále). Relace a entity logický model reprezentuje jako množinu relačních tabulek. Tato fáze návrhu slouží jako zdroj informací a dat pro fyzickou fázi návrhu. Umožňuje návrháři databáze uvažovat nad alternativními postupy při fyzické realizaci databáze, to je pro efektivní návrh databáze velmi důležité [9].

Ve fyzické fázi návrhu databáze se návrhář rozhoduje jak fyzicky implementovat logický návrh v prostředí cílového systému řízení báze dat. Tato fáze konkrétně upravuje návrh databáze pro specifický databázový systém (SQL Server, Access, Oracle) [9].

### 1.8.2 Zásady pro úspěšný návrh databáze

Existují zásady důležité pro úspěšnost návrhu databáze:

- Interaktivní práce s uživateli co nejvíce je to možné
- Postup dle strukturované metodologie během celého procesu návrhu
- Využívání daty řízeného přístupu
- Zahrnutí struktury a integrity do datového modelu
- Použití normalizace a techniky ověření správnosti transakcí v metodologii
- Použití diagramu pro reprezentaci modelu dat
- Používání jazyku pro návrh databáze
- Vytvoření slovníku dat

Všechny výše uvedené zásady jsou obsaženy v konceptuální metodologii [9].

### 1.8.3 Konceptuální návrh databáze

Konceptuální návrh databáze je první fáze konceptuální metodologie návrhu databáze. Jedná se o konceptuální datový model, který popisuje obsah dat systému na úrovni, která je nezávislá na vlastním implementačním a technologickém prostředí. Konceptuální návrh databáze plní následující funkce: poznávání zkoumané části reality, komunikace mezi členy týmu (řešitelského), platforma pro diskuse s uživateli, podklad pro návrh datové základny, dokumentace existující datové základny [10].

Konceptuální návrh databáze vytváří ER (entitně-relační) model, který představuje úplnou a přesnou reprezentaci datových požadavků organizace nebo její části, kterou má databáze podporovat. Model zajišťuje minimální redundanci dat a je schopen podporovat transakce. Každý ER model obsahuje entity, relace, atributy a jejich domény, klíče kandidátní, primární a integritní omezení. ER model je popsán v dokumentaci, ve které jsou zahrnuty i ER diagramy [9].

Konceptuální model se dělí na dvě podúrovně: konceptuální schéma a konceptuální model dat. Konceptuální schéma zachycuje základní entitní množiny, vztahy a jejich atributy. Konceptuální model dat popisuje obsah datové základny a je vytvořen za účelem přesného obsahu datové základny [10].

### 1.8.4 Logický návrh databáze

Logický návrh databáze převádí ER model získaný v konceptuálním návrhu databáze do množiny relačních tabulek. Při tvorbě logického návrhu databáze se zabýváme otázkou „co“. Pro minimalizování redundance se kontroluje, zda struktura tabulek vyhovuje normalizačním formám. Definují se požadovaná integritní omezení a zkontroluje se možnost provádění požadovaných transakcí nad tabulkami. Logický návrh je nezávislý na databázovém systému řízení dat i na fyzické implementaci [9].

Díky definicím požadovaných integritních omezení se zabrání, aby se databáze stala nekompletní, nepřesnou či nekonzistentní. Tato pravidla zahrnují požadavky na data, omezení domén sloupců, integritu dat, multiplicitu, referenční integritu a další integritní omezení [9].

### 1.8.5 Fyzický návrh databáze

Fyzický návrh databáze navazuje na logický návrh databáze, který je pro něj zdrojem informací. Jedná se o proces vytvoření popisu implementace databáze. Popisuje podkladové tabulky, organizaci souborů, používané indexy (pro efektivní přístup k datům), další integritní omezení a bezpečnostní omezení. Zabývá se otázkou „jak“. Návrhář databáze musí v této fázi znát počítačový systém a funkčnost cílového databázového systému řízení dat (protože jednotlivé DBMS se svou funkčností v současné době odlišují). Funguje zde i zpětná vazba – jsou-li učiněna nějaké rozhodnutí týkající se například zlepšení výkonnosti, pak to ovlivní i logický návrh databáze [9].

## 1.9 Indexy

Index je pomocná datová struktura, která umožňuje rychlejší lokalizaci konkrétních záznamů v souboru pro systém řízení databáze, a tím zrychluje odezvu na dotazy uživatelů. Indexy v databázi lze srovnávat s rejstříkem v knize. Indexy nejsou pro používání nezbytné, ale mohou mít podstatný vliv na výkonnost databáze [9]. Indexy jsou nejjednodušeji upravovatelnými strukturami databáze a nevyžadují změny v aplikacích. [3]. Indexy by ale neměly být používány nadbytečně, protože zpomalují operace typu INSERT, UPDATE a DELETE. V databázi je možné mít index na více sloupcích tabulky najednou – nazývají se vícesloupcové indexy. Tyto vícesloupcové indexy mají reálný význam pouze v případě, že vyhledáváme data pomocí hodnot v indexovaných sloupcích. Je možné mít i více indexů na jedné tabulce, pokud se týkají jiných sloupců [8]. Indexy můžeme vytvořit při definici tabulky, ale není to nutné. Je možné a relativně jednoduché přidat je do databáze, i po zavedení do provozu, beze změn kódu, ale generování indexů u velkých tabulek může být časově náročné. Návrhář či správce databáze musí neustále zkoumat databázi a indexu. Musí být neustále připraven provádět případné potřebné změny [3].

Struktura indexu souvisí s vyhledávacím klíčem a obsahuje záznamy o hodnotě klíče a adrese logického záznamu v souboru. Soubor s logickými záznamy je označován jako datový soubor a soubor se záznamy o indexu je označován jako indexový soubor. Hodnoty v indexovém souboru jsou řazeny podle indexačního pole (jeden sloupec). Mezi nejvýznamnější typy patří primární index, seskupovací index a sekundární index. Primární index má datový soubor sekvenčně řazený podle klíčového pole a indexační pole je založeno na seřazeném klíčovém poli. U klíčového pole je záruka jedinečné hodnoty každého záznamu. U seskupovacího indexu je datové pole řazeno podle neklíčového pole (nazývané také seskupovací pole) a indexační pole je založeno na tomto neklíčovém poli. Každé hodnotě indexu může tedy odpovídat více záznamů v datovém souboru. Sekundární index je definovaný na základě nesetříděného pole datového souboru. Soubor může mít nejvýše jeden primární index, jeden seskupovací index a navíc může mít několik sekundárních indexů. Pro vytváření indexů se používá příkaz CREATE INDEX [9].

## 1.10 Uložené procedury

Uložené procedury mají několik výhod, ale mají i své nevýhody. Mezi výhody patří především to, že izolují aplikaci od změn v databázi. Pokud se tedy změní struktura databáze, stačí modifikovat uložené procedury a aplikace jako takové se změny nedotknou. To se týká ale pouze takových změn, které zachovávají logický význam uložených dat, a těch nebývá mnoho. Přibydou-li například nové sloupce nebo tabulky, aplikace využívající data z databáze musí na tuto změnu reagovat. Další výhoda má spočívat ve vyšší výkonnosti databáze. Dochází k tomu zejména tehdy, pokud procedura obsahuje více příkazů a porovnáváme ji se situací, kdy posíláme tyto příkazy na server postupně ve více dávkách. Pokud příkazy na server pošleme v jedné dávce, nebude rozdíl tak významný. Kód uložené procedury bude zkompilován a zoptimalizován pouze jednou a proveden může být mnohokrát. To má důležitou roli, pokud procedura obsahuje složitější příkaz nebo příkazy, v tomto případě bývá uložená procedura rychlejší. Naopak bude-li SQL příkaz jednoduchý, může být přímo odeslaný příkaz na server rychlejší, protože optimalizace příkazu nezabere tolik času jako volání procedury [5].

Uložené procedury mají i jisté nevýhody. Nevýhodou důsledného používání uložených procedur může být náročnější správa aplikace. Jsou totiž tři oblasti, které musí být kompletně synchronizovány: datový model, uložené procedury a vlastní kód aplikace. Změna v datovém modelu vyžaduje modifikaci na uložených procedurách i ve vlastním kódu aplikace. Omezíme-li uložené procedury, o něco se zjednoduší i správa aplikace [5].

## 1.11 Zabezpečení databáze

Bezpečnost by měla být první myšlenka, která napadne správce MySQL databáze po vytvoření nové databáze. Zabezpečení databáze brání odcizení či úmyslnému poškození dat někým nepovolaným. Při zabezpečování databáze lze využít možností správy přístupů – to znamená, že správce databáze přiřadí jednotlivým uživatelům oprávnění [2].

## 1.12 Provoz a údržba databáze

Problém, se kterým se můžeme setkat už v době vývoje databáze, spočívá v udržování verzí databáze. Tento problém nastává, když na vývoji databáze pracuje několik vývojářů nebo pokud je u nějakého klienta spuštěna nějaká verze aplikace. Abychom se vyhnuli tomuto problému, je třeba udržovat historii verzí, nejlépe i s testovacími daty [5].

Od okamžiku kdy je první verze distribuována k uživateli, je potřeba postupovat obezřetně. Pro uživatele data uložená v databázi znamenají jistou hodnotu. Proto s novou verzí distribuovanou k uživateli musíme provést automatický upgrade struktury databáze [5].

U větších databází je v rámci údržby užitečné aktualizovat například interní statistiky, přegenerování indexů a podobné úkoly, které zvyšují výkonnost databáze. Tyto akce by měly probíhat v pravidelných intervalech tak, aby nezatěžovaly uživatele [5].

## 1.13 Zálohování databáze

Pro zálohování dat z databáze je nutné připravit si způsob, jak budeme tato data zálohovat. Zálohování dat má na starosti buď databázový administrátor, nebo aplikace. Má-li na starosti zálohování dat aplikace je nudné aby aplikace uživateli připomínat nutnost zálohování dat zálohování dat aplikace je nudné aby aplikace uživateli připomínat nutnost zálohování dat, nebo si zálohování dat vynutit, protože uživatelé na zálohování dat budou často zapomínat. Při vývoji by se měla nasimulovat havárie a vyzkoušet zda je data možné ze zálohy bez problému obnovit [5].

# 2 ANALÝZA SOUČASNÉHO STAVU

Společnost ELTEX electronics, pro kterou je tato práce zpracovávána, sídlí v obci Bezměrov – okres Kroměříž. Společnost se zabývá vývojem a konstrukcí tepelných čerpadel a prodejem dílů potřebných pro stavbu tepelných čerpadel. Nabízí neúplná tepelná čerpadla jako tzv. stavebnice, kompletní tepelná čerpadla, ale i možnost zapojení čerpadla do konstrukce klienta či využití dílů dodaných klientem do stavby tepelného čerpadla.

Na počátku měla společnost jednu počítačovou stanici, na které byla naistalována měřící aplikace komunikující s měřicím přístrojem, či několika měřicími přístroji připojenými na tuto počítačovou stanici. Měřící aplikace ukládala naměřená data na lokální disk počítačové stanice. S rozvojem společnosti a její činnosti přibývalo měřicích přístrojů i počítačových stanic. S nárůstem počtu měřicích přístrojů a počítačových stanic přibyla i potřeba všechna tato data centralizovat na jedno úložiště a mít tak možnost tato data vizualizovat například v jednom grafu.

Společnost si na navrhovanou databázi zadala několik požadavků. Požadavky se týkají především přizpůsobení návrhu databáze pro následnou vizualizaci dat, kterou bude zajišťovat aplikace. Další požadavky společnosti se týkají měřených dat. Požadavky budou podrobněji popsány níže.

## 2.1 Požadavky společnosti

V této kapitole práce budou blíže popsány požadavky společnosti ELTEX electronics na navrhovanou databázi. Požadavky na navrhovanou databázi budou rozděleny do dvou subkapitol. První subkapitola je zaměřena na požadavky společnosti týkající se přizpůsobení databáze pro následnou vizualizaci dat a druhá subkapitola je zaměřena na požadavky společnosti na archivaci měřených dat.

### 2.1.1 Vizualizace dat

Databáze bude obsahovat aktuální naměřené hodnoty získávané z měřicích přístrojů v intervalu několika sekund a archivní data, která budou ukládána dle nastavení v měřící aplikaci. Výchozí nastavení pro archivaci dat bude jedna minuta. Aktuální hodnoty i archivní data budou neustále k dispozici pro zobrazení v klientské aplikaci

na libovolném počítači v síti. Aktuální hodnoty budou zobrazovány jako číselné hodnoty, archivní data budou zobrazována v grafech. Klientská aplikace bude umožňovat definování pohledu na data, jehož nastavení bude uloženo v databázi tak, aby tento pohled bylo možné zobrazit na více počítačích v klientské aplikaci současně. Pro každou jednotku archivních dat v grafech bude možné nastavit rozsah hodnot zvlášť. Toto nastavení bude také uloženo v databázi, aby bylo zajištěno totožné zobrazení grafů ve všech klientských aplikacích na počítačích v síti.

Samotnou vizualizaci dat z databáze bude zajišťovat aplikace, která bude s databází komunikovat. Tuto aplikaci si společnost zpracuje sama v návaznosti na návrh databáze zpracovaný v předkládané práci.

### 2.1.2 Měřená data

Ke každé měřící stanici může být připojen jeden nebo více měřicích přístrojů. Každý připojený měřicí přístroj může měřit jednu či více hodnot, které se budou v databázi uchovávat. Měřené hodnoty budou jak analogové (například °C, kW, kWh), tak i digitální. Jednotky měřených hodnot budou definované v aplikaci v závislosti na měřicích přístrojích.

Protože se společnost zabývá i vývojem a konstrukcí tepelných čerpadel, potřebuje, aby bylo možné připojit a v databázi nadefinovat novou jednotku, pro případ připojení nového typu měřicího přístroje.

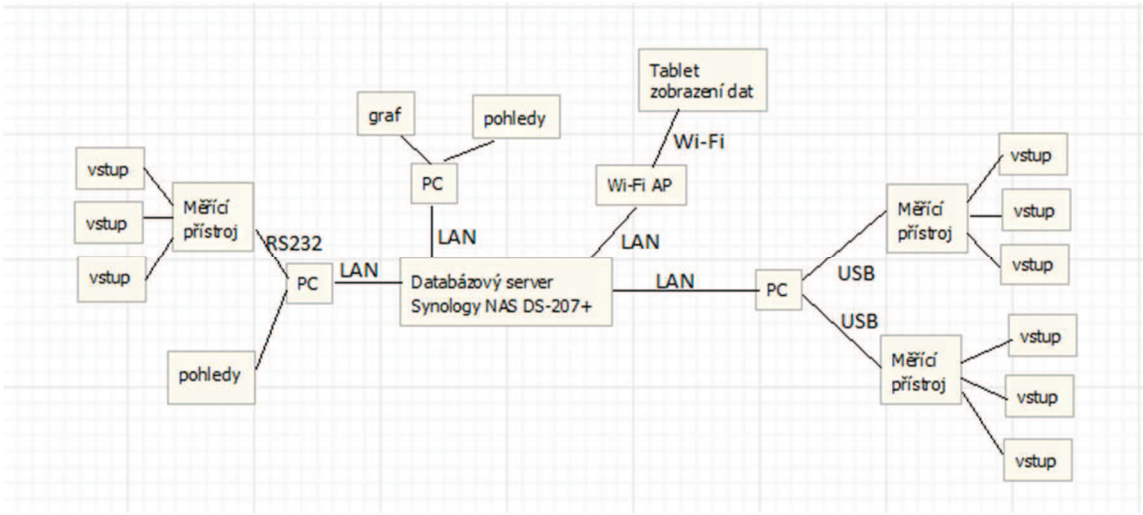
## 2.2 Síťová infrastruktura

Společnost má již připravenou síťovou infrastrukturu s měřícími počítačovými stanicemi, měřicími přístroji i s databázovým serverem Synology NAS DS-207+. Databázový server Synology NAS DS-207+ má dva terabajtové disky se zabezpečením dat RAID 1. Zabezpečení RAID 1 představuje zrcadlové ukládání dat na oba dva disky. Jedná se o nejjednodušší ale poměrně efektivní ochranu dat. V případě výpadku jednoho disku server pracuje s kopií, která je ihned k dispozici na druhém disku.

Na databázový server Synology NAS DS-207+ jsou připojené počítačové stanice pomocí sítě LAN (Local Area Network), tato část LAN sítě je tvořena technologií Ethernet. LAN je lokální síť a v tomto případě pokrývá pouze prostory společnosti ELTEX electronics. Všeobecně se LAN sítě vyznačují vysokou rychlostí. Databázový server komunikuje

prostřednictvím LAN sítě s místním Wi-Fi routrem, který je v síťové infrastruktuře umístěn z důvodu připojování přenosných zařízení sloužících k vizualizaci dat (například grafů) z klientské aplikace, jako jsou tablety, chytré mobilní telefony nebo notebooky. Na počítačové stanice jsou připojené sběrnicí USB (Universal Seriál Bus) nebo sériovou linkou RS-232 měřicí přístroje. Sériová linka RS-232 se využívá jako komunikační rozhraní osobních počítačů a dalších elektronických zařízení. Sběrnice USB je univerzální sběrnicí a připojuje periferní zařízení k osobním počítačům. Sběrnice USB má jen jedno zařízení typu master. Jen zařízení typu master může vysílat samo od sebe, ostatní zařízení připojená na tuto sběrnici mohou vysílat až po přijetí požadavku na vysílání dat od zařízení typu master. Počítačové stanice mohou sloužit i k vizualizaci dat prostřednictvím klientské aplikace. Každý měřicí přístroj může mít jeden nebo více vstupů.

Pro představu, jak celá připravená síťová infrastruktura vypadá a funguje, slouží následující schéma síťové infrastruktury. Schéma síťové infrastruktury je ve větší velikosti v příloze práce.




Obrázek 2: Schéma síťové infrastruktury

# 3 VLASTNÍ NÁVRHY ŘEŠENÍ

Tato část práce je zaměřena na vlastní zpracování návrhu databáze pro společnost ELTEX electronics. Databáze je navrhována v MySQL, protože společnost ELTEX electronics má připravený databázový server i s databázovým systémem MySQL. Do návrhu databáze jsou zapracovány požadavky společnosti. Zpracování návrhu databáze bylo provedeno v programu MySQL Workbench verze 6.0 CE z důvodu snadného grafického zpracování návrhu. Takovýto návrh databáze je přehledný a je jednodušší se v něm zorientovat. MySQL Workbench je dostupný ke stažení z: http://dev.mysql.com/downloads/tools/workbench/.

Pro zpracování návrhu databáze bude postupováno dle konceptuální metodologie, která je popsána v teoretické části této práce. Konceptuální metodologie má tři fáze: konceptuální fáze, logická fáze a fyzická fáze.

## 3.1 MySQL Workbench

Jedná se o program nejen pro vizuální návrh struktury databáze MySQL. Je vydáván pod licencí GPL (General Public License), jedná se tedy o open source software. Tento software je plně funkční na všech nejrozšířenějších platformách, jako je Windows, Linux a OS X.

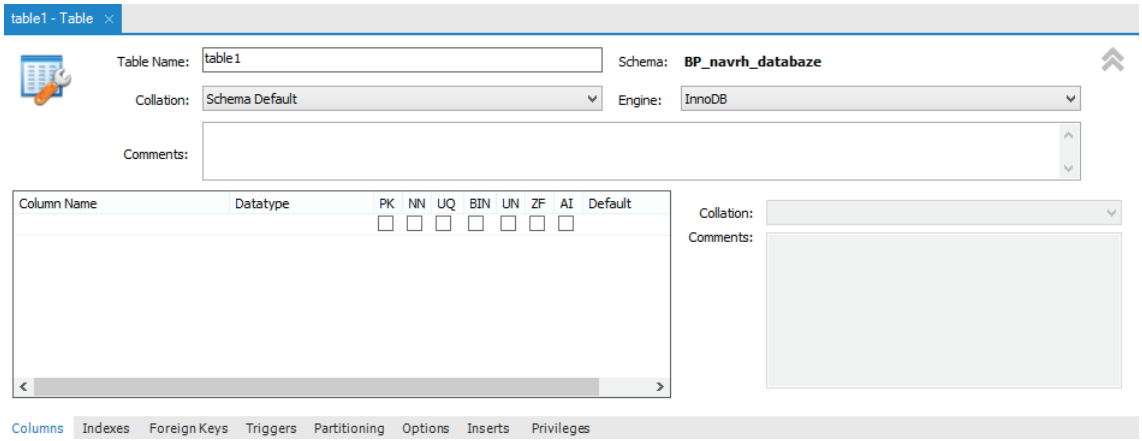
Po nainstalování a spuštění MySQL Workbench 6.0 jsem vytvořila nový model, tento model je možné přejmenovat a nastavit kódovací jazyk pro tento model. Jako výchozí kódovací jazyk je nastaven UTF8, který jsem pro svůj návrh ponechala. Dále jsou na úvodní obrazovce ikony pro tvorbu nové tabulky, pohledu a procedury. Je možné definovat i uživatele a jejich role. Jednotlivým uživatelům/rolím lze nastavovat pravomoci a odpovědnosti. V MySQL Workbench lze psát i přímo SQL skript. SQL skript lze ale vygenerovat na základě vytvořeného ER diagramu.

### 3.1.1 Nová tabulka a její atributy

Při tvorbě nové tabulky (entity) lze nastavit název tabulky, typ úložiště (engine), komentář a vytvářet jednotlivé atributy tabulky. U atributů lze nastavovat parametry: primary key - PK (primární klíč), not null - NN (nenulové hodnoty), unique index - UQ

(unikátní index), binary column -BIN (binární atribut), unsigned data type -UN (pouze kladné hodnoty čísla) a auto incremental - AI (automatické číslování).

Pokud použijeme parametr unsigned data type, pak bude datový typ bez znaménka (pouze kladná čísla). Jedná-li se o celočíselný datový typ, pak se posune interval hodnot, jedná-li se o datový typ s pohyblivou desetinnou čárkou, pak se interval neposouvá, ale berou se jen kladné hodnoty. Posuny intervalů hodnot jsou uvedené v teoretické části v kapitole datové typy.




Obrázek 3: MySQL Workbench - nová tabulka

## 3.2 Fáze konceptuálního návrhu

V této kapitole bude popsána fáze konceptuálního návrhu z konceptuální metodologie konkretizována na zpracovávaný návrh databáze. Tato fáze se zaměřuje především na identifikaci entit a relací mezi nimi.

### 3.2.1 Návrh entit

V navrhované databázi pro společnost ELTEX electronics bude navrženo celkem čtrnáct entit (tabulek) v entito-relačním modelu. Entity budou pojmenovávány systematicky, podle toho o čem mají uchovávat data.

Následující seznam entit obsahuje názvy i stručný popis k čemu jsou určené:

- Merici_stanice – uchovává informace o jednotlivých počítačových měřících stanicích v síťové infrastruktuře
- Merici_pristroj – entita s informacemi a vlastnostmi každého použitého měřicího přístroje připojeného do síťové infrastruktury

- Vstup – tabulka jednotlivých vstupů měřicích přístrojů
- Hodnota – tato entita uchovává naměřené hodnoty z měřicích přístrojů
- Cas – entita sloužící k ukládání časových razítek. Je vyčleněna, aby byly splněny podmínky normalizace a aby bylo ušetřené místo na serverových discích. V jeden moment bude uloženo několik záznamů.
- Typ_pristroje – entita definující typ měřicího přístroje zapojeného do síťové infrastruktury. Několik přístrojů může být stejného typu, proto jsou tato data vyčleněna do zvláštní entity.
- Jednotka – tabulka jednotek měřených hodnot. Vyčleněna z důvodu úspory místa na discích a pro splnění podmínek normalizace.
- Aktualni_hodnota – entita uchovávající pouze nejaktuálnější hodnoty získané z měřicích přístrojů. V této tabulce budou data neustále přepisována.
- Rozsah – slouží k uložení definovaného rozsahu jednotky zobrazovací aplikací. Rozsah jednotek bude možné nastavit pro každou jednotku zvlášť pro zobrazení archivních dat v grafech.
- Nastaveni_grafu – informace z této entity definují pozici a velikost zobrazovaného grafu – aby bylo možné zobrazovat graf na stejném místě ať už opakovaně na jednom zobrazovacím zařízení nebo na více zobrazovacích zařízeních v síťové infrastruktuře zároveň.
- Pohled_vstup – dekompoziční entita relace M:N mezi tabulkami pohled a vstup.
- Nastaveni_zobrazeni – entita uchovávající informace o nastavení zobrazení pohledu, o umístění a velikosti pohledu na zobrazovacím zařízení
- Pohled – entita s informacemi, které vstupy se mají zobrazovat v jakém pohledu aplikace.
- Skupina_pohledu. – tato entita je určena pro ukládání informací o tom, které pohledy jsou zobrazené v jaké skupině. Tyto informace slouží k jednotnému zobrazení pohledů a grafů na více zařízeních v síti zároveň.

Většina navrhovaných entit bude typu datového úložiště MyISAM, jen tabulka aktualni_hodnota bude typu MEMORY. Typ datového úložiště MEMORY je zvolen pro

tuto tabulku pro jeho extrémní rychlost, protože v tabulce aktualni_hodnota je potřeba rychle přepisovat data aktuálně naměřenými hodnotami a také je potřeba rychle tyto hodnoty číst.

### 3.2.2 Návrh relací

Mezi většinou entit je vztah 1:N, který umožňuje jednomu záznamu z jedné entity přiřadit více záznamů z jiné entity. Entita merici_stanice je ve vztahu 1:N s entitou merici_pristroj, což značí, že na jednu měřící stanici může být připojen 1 nebo více měřicích přístrojů.

Entita merici_pristroj je ve vztahu N:1 s entitou typ_pristroje, tento vztah znamená, že jeden či více přístrojů je stejného typu. S tabulkou merici_pristroj má vztah ještě tabulka vstup a to 1:N, to znamená, že jeden měřicí přístroj může mít jeden nebo více vstupů (tj. jeden měřicí přístroj může měřit více hodnot).

Tabulka vstup je ve vztahu celkem s pěti tabulkami. Vztah tabulky vstup a tabulky merici_pristroj je již popsán výše. Entita vstup je ve vztahu 1:N pouze s entitou hodnota – jednomu vstupu náleží více hodnot (časově rozlišených). S dalšími entitami je ve vztahu N:1, jedná se o entitu jednotka a propojovací tabulku pohled_vstup. Tyto dva vztahy znamenají, že více vstupů může mít stejnou jednotku a že jeden záznam z propojovací tabulky pohled_vstup odpovídá více záznamům v tabulce vstup. Pátý vztah tabulky vstup je 1:1 a bude popsán níže.

Entita hodnota je ve vztahu ještě s entitou cas. Tento vztah je N:1 a to znamená, že více hodnot může mít stejný čas. To je dáno periodickým ukládáním naměřených hodnot – a z toho důvodu byl čas vyčleněn do samostatné entity.

Entita jednotka je ve vztahu ještě s entitou rozsah. Tyto dvě entity mezi sebou mají vztah 1:N – jedna jednotka může mít nastavených více rozsahů (pro jednotlivé grafy).

Propojovací tabulka pohled_vstup je ve vztahu celkem se třemi entitami. Vztah s entitou vsup je popsán výše. S dalšími dvěmi entitami je ve vztahu 1:N, jedná se o entity nastaveni_zobrazeni a pohled. Tyto dva vztahy značí, že jednomu záznamu z tabulky pohled_vstup odpovídá více záznamů z tabulky pohled (jedna položka může být zobrazena ve více pohledech) a z tabulky nastaveni_zobrazeni (pro jednotlivé pohledy má položka různé nastavení).

Entita pohled je ve vztahu se čtyřmi entitami, jeden z těchto vztahů (vztah s entitou pohled_vstup) je popsán výše. S dalšími dvěmi entitami je ve vztahu 1:N, jde o entity nastaveni_zobrazeni a nastaveni_grafu. Jeden pohled tedy může mít více nastavení zobrazení i více nastavení grafu. S poslední entitou je entita pohled ve vztahu N:1 – jedná se o vztah s entitou skupina_pohledu, což znamená, že v jedné skupině pohledů může být jeden či více pohledů.

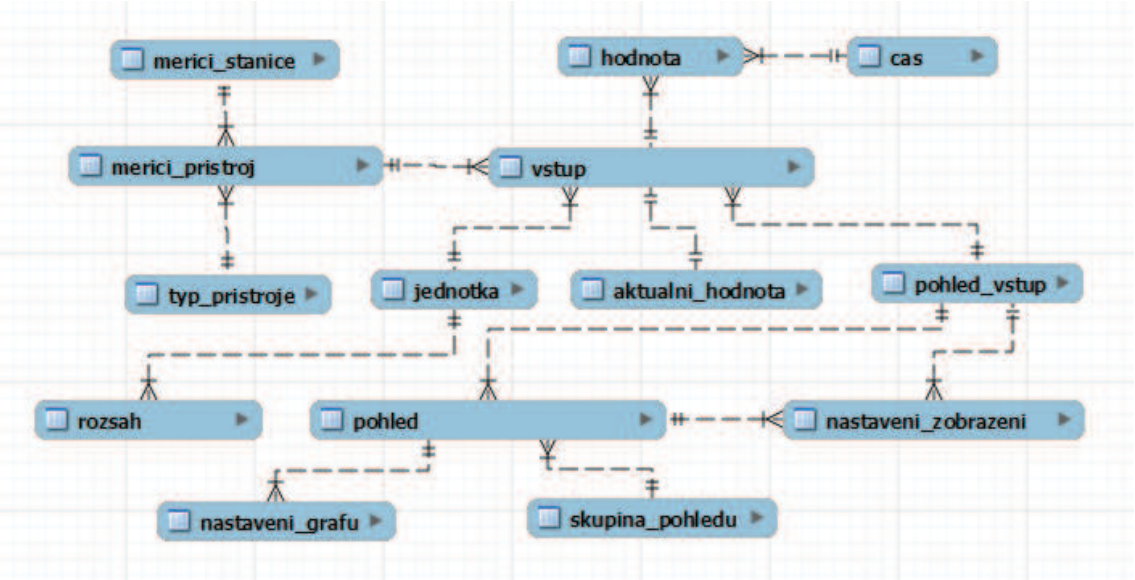
V návrhu je využit i vztah 1:1 a to mezi entitami vstup a aktualni_hodnota. Vztah 1:1 přiřazuje jednomu záznamu z první entity právě jeden záznam z druhé entity. Tento vztah byl zvolen mezi těmito entitami z důvodu potřeby rychlého přepisování hodnot v tabulce aktualni_hodnota na ty nejaktuálněji naměřené hodnoty. V tabulce aktualni_hodnota budou tedy dostupné pouze nejaktuálnější naměřené hodnoty ze všech aktivních měřicích přístrojů. Pro archivaci naměřených hodnot slouží tabulka hodnota. Do tabulky hodnot se budou zapisovat data periodicky. Perioda pro ukládání dat bude definována v nastavení aplikace. Tato archivní data budou využívána jako zdrojová data pro následnou vizualizaci.

### 3.2.3 Shrnutí fáze konceptuálního návrhu

V této fázi návrhu databáze byly definovány veškeré entity, které budou použity v návrhu databáze. Celkem bylo tedy navrženo 14 entit. Následně bylo nadefinováno, jakého typu datového úložiště tyto entity budou. Posledním krokem fáze konceptuálního návrhu byl návrh vztahů mezi jednotlivými entitami. Grafické znázornění vztahů mezi entitami je uvedeno v kapitole fáze logického návrhu, která se zabývá převedením konceptuálního návrhu do množiny relačních tabulek a dalším zpracováním informací z konceptuálního návrhu.

## 3.3 Fáze logického návrhu

Logický návrh databáze se zabývá převedením ER modelu z konceptuálního návrhu databáze do množiny relačních tabulek. Logický návrh navrhované databáze je zpracován do schématu, pro přehlednost a snadnou orientaci. Schéma navrhnutých entit a relací mezi nimi:



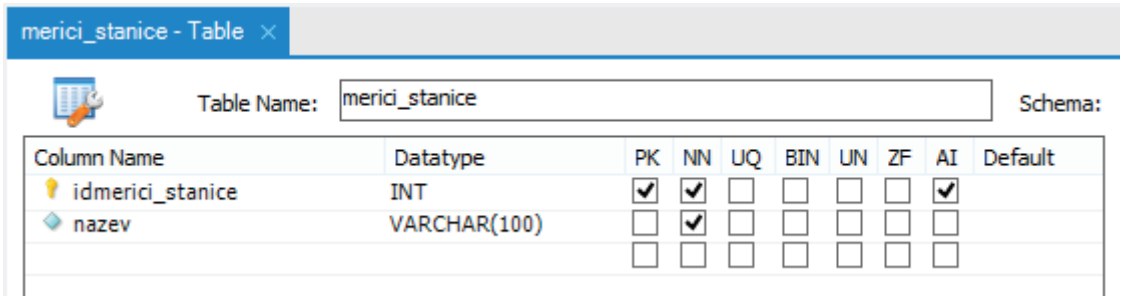

Obrázek 4: MySQL Workbench - diagram entit a vztahů mezi nimi

Dále se logický návrh zabývá, zda struktura tabulek odpovídá normalizačním formám, proto budou dále v této kapitole detailněji popsány jednotlivé atributy entit celého návrhu databáze.

Jako primární klíče entit budou sloužit identifikační čísla z důvodu jednoduchosti a unikátnosti, což jsou důležité vlastnosti primárního klíče. Vztahy mezi jednotlivými entitami budou zajištěny pomocí primárních a cizích klíčů entit. Cizí klíče jednotlivých entit v práci budou popisovány systematicky podle vztahů mezi entitami dle výše uvedeného diagramu entit a vztahů mezi nimi. Pojmenování cizích klíčů lze popsat i následujícím systémem: „jméno tabulky“_“primární klíč tabulky“.

### 3.3.1 Měřící stanice

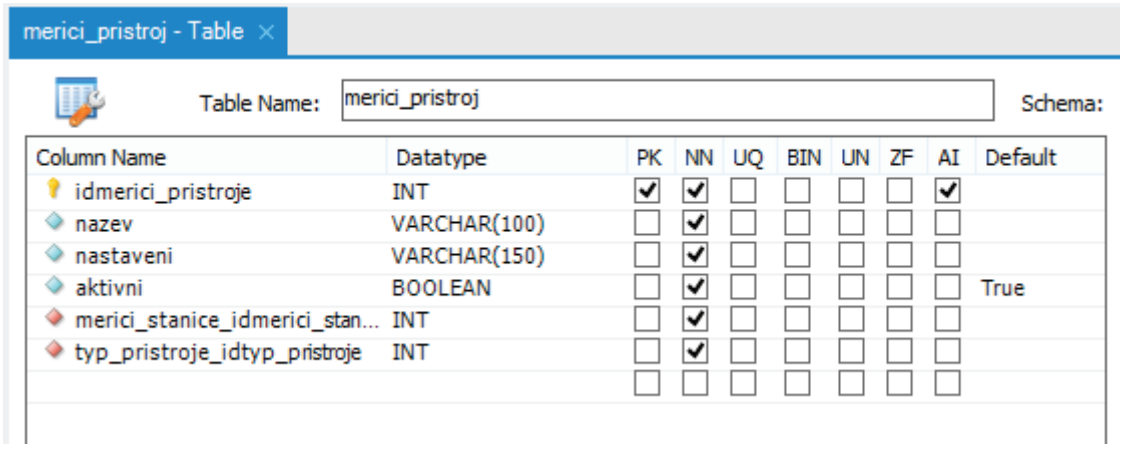
Entita měřící stanice bude uchovávat informace o všech použitých měřících stanicích. Bude obsahovat následující dva následující atributy: identifikační číslo měřící stanice (idmerici_stanice) a název měřící stanice (nazev). Identifikační číslo měřící stanice je použito zároveň jako primární klíč entity a má nastavené automatické číslování. Oba atributy této entity mají nastavený parametr „not null“. Na tuto entitu je navázána entita merici_pristroj vztahem 1:N.




Obrázek 5: MySQL Workbench - atributy entity merici_stanice

### 3.3.2 Měřicí přístroj

Tato entita obsahuje informace o všech využitých měřicích přístrojích v síti. Informace jsou uchovávány v atributech: identifikační číslo měřicího přístroje (idmerici_pristroj), název měřicího přístroje (nazev), nastavení měřicího přístroje (nastaveni), informaci zda je daný měřicí přístroj aktivní nebo pasivní (aktivni), a dva cizí klíče pro vztah s entitou měřící stanice (merici_stanice_idmerici_stanice) a pro vztah s entitou typ přístroje (typ_pristroje_idtyp_pristroje). Tyto dvě navázané entity jsou ve vztahu N:1. Entita měřicí přístroj má vazbu ještě s entitou vstup, se kterou je ve vztahu 1:N. Všechny atributy entity měřicí přístroj mají nastavený parametr „not null“. Identifikační číslo měřicího přístroje je primárním klíčem entity, a proto má nastavené automatické číslování. Výchozí hodnota atributu aktivni je nastavena na „True“.

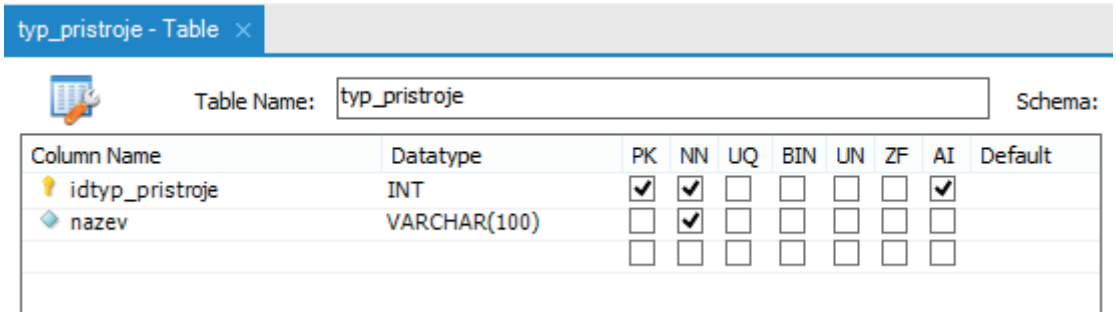



Obrázek 6: MySQL Workbench - atributy entity merici_pristroj

### 3.3.3 Typ přístroje

Entita typ_pristroje obsahuje identifikační číslo typu přístroje (idtyp_pristroje) a název typu přístroje (nazev). Identifikační číslo typu přístroje slouží jako primární klíč

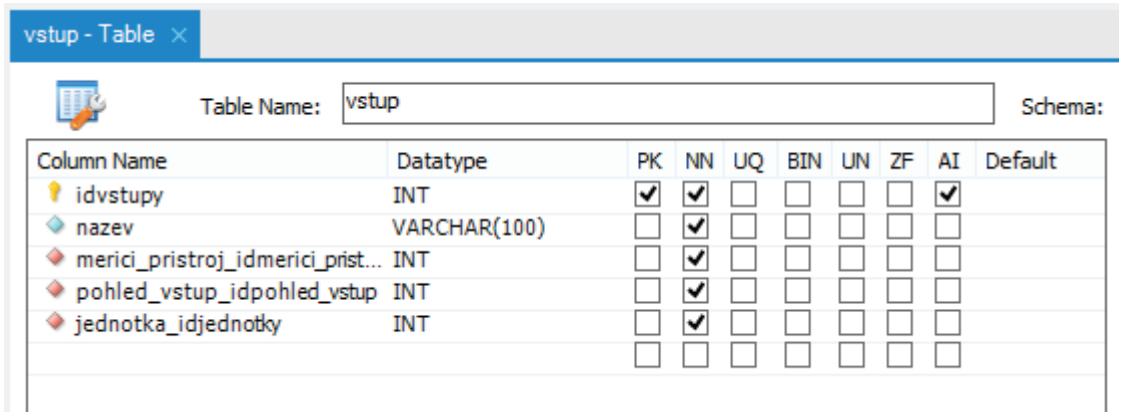
a má nastavené automatické číslování. Oba atributy této entity mají nastavené kritérium „not null“. Na tuto entitu je navázána entita merici_pristroj vztahem 1:N.




Obrázek 7: MySQL Workbench - atributy entity typ_pristroje

### 3.3.4 Vstup

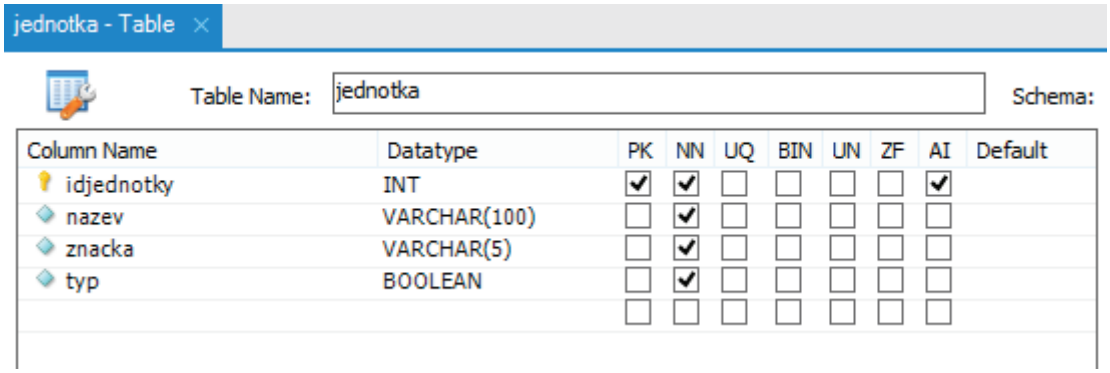
Entita vstup obsahuje identifikační číslo vstupu (idvstup), název vstupu (nazev) a tři cizí klíče. Entita vstup je celkem provázána s pěti tabulkami. První cizí klíč je identifikační číslo měřicího přístroje (merici_pristroje_idmerici_pristroje), který vytváří vztah mezi entitou vstup a entitou měřicí přístroj. Druhý cizí klíč je identifikační číslo jednotky (jednotky_idjednotky), tento cizí klíč vytváří vazbu mezi entitou vstup a entitou jednotka. Třetím cizím klíčem je identifikační číslo entity pohled_vstup (pohled_vstup_idpohled_vstup), tento cizí klíč vytváří propojení mezi entitou vstup a entitou vstup_pohled. Entita vstup_pohled je dekompoziční tabulkou mezi entitami vstup a pohled, mezi kterými by byla vazba M:N. Ve všech případech jsou entity s entitou vstup ve vztahu 1:N, kdy N je na straně entity vstup. Identifikační číslo vstupu slouží jako primární klíč a má navíc oproti ostatním atributům nastavené automatické číslování. Všechny atributy této tabulky mají nastavený parametr „not null”.




Obrázek 8: MySQL Workbench - atributy entity vstup

### 3.3.5 Jednotka

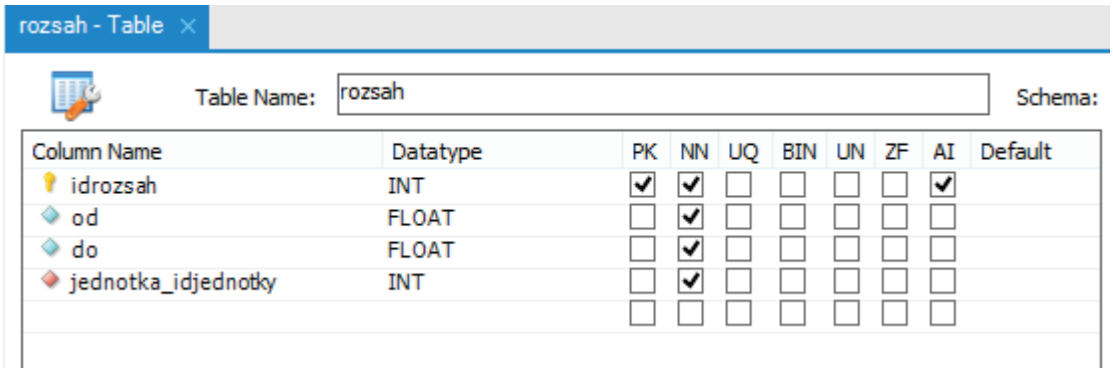
Entita jednotka obsahuje identifikační číslo jednotky (idjednotka), název jednotky (nazev), značku jednotky (znacka), typ jednotky (typ). Identifikační číslo jednotky je nastavené jako primární klíč a proto má, tak jako ostatní primární klíče ostatních entit, nastavené automatické číslování. Typ jednotky udává, zda se jedná o jednotku analogovou či digitální, z tohoto důvodu je datového typu boolean. U všech atributů této entity je nastavený parametr „not null“. Tabulka jednotky je navázána na další dvě entity. S oběma navázanými entitami má vztah 1:N. Vztahy má entita jednotka s entitami rozsah a vstup. Vazby s těmito entitami jsou zajištěny cizími klíči odkazující se na primární klíč entity jednotka.




Obrázek 9: MySQL Workbench - atributy entity jednotka

### 3.3.6 Rozsah

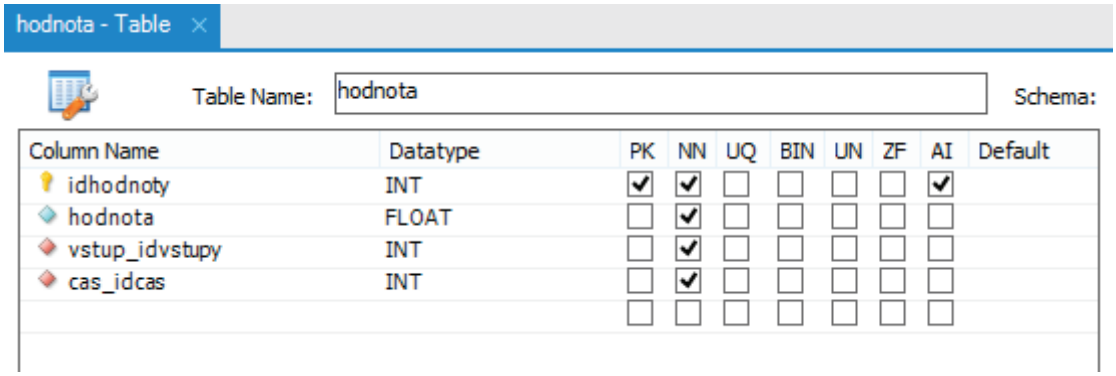
Pro tuto entitu je primárním klíčem identifikační číslo (idrozsah), které má nastavené automatické číslování. Tabulka rozsah obsahuje dva atributy, které slouží k definování rozsahu jednotky v grafech a jsou pojmenovány od a do. Dále obsahuje cizí klíč jednotka_idjednotka, který slouží k provázání s tabulkou jednotka, tyto dvě entity jsou ve vztahu N:1. Všechny atributy této tabulky mají nastavený parametr „not null“.




Obrázek 10: MySQL Workbench - atributy entity rozsah

### 3.3.7 Hodnota

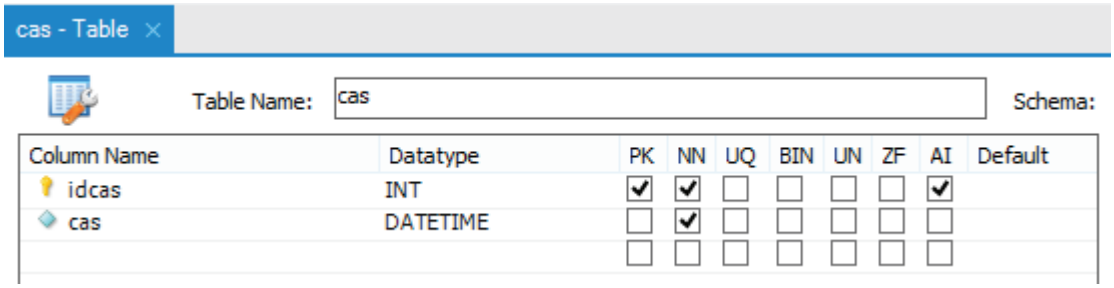
Entita hodnota je navrhnuta pro uchovávání naměřených hodnot z měřicích přístrojů. Data do této entity budou zapisována periodicky podle nastavení v ovládací aplikaci. Entita obsahuje pouze atribut hodnota a atribut identifikační číslo hodnoty (id_hodnota), který slouží jako primární klíč. Tento primární klíč má, jako ostatní primární klíče v tomto návrhu databáze, nastavené automatické číslování. Dále obsahuje identifikační čísla tvořící vazby na entity čas a vstup. S těmito dvěma entitami má tabulka hodnota vazbu N:1. Všechny atributy tabulky mají také nastavený parametr „not null“.




Obrázek 11: MySQL Workbench - atributy entity hodnota

### 3.3.8 Čas

Tabulka čas obsahuje pouze identifikační číslo (idcas) a atribut čas (cas). Identifikační číslo je primárním klíčem a má nastavené automatické číslování. Atribut čas je datového typu datetime. Na tuto tabulku čas je navázána pouze tabulka hodnota, se kterou má entita čas vztah 1:N. Tabulka čas je z tabulky hodnota vyčleněna z důvodu úspory místa v databázi, protože v jeden čas bude ukládáno více hodnot.

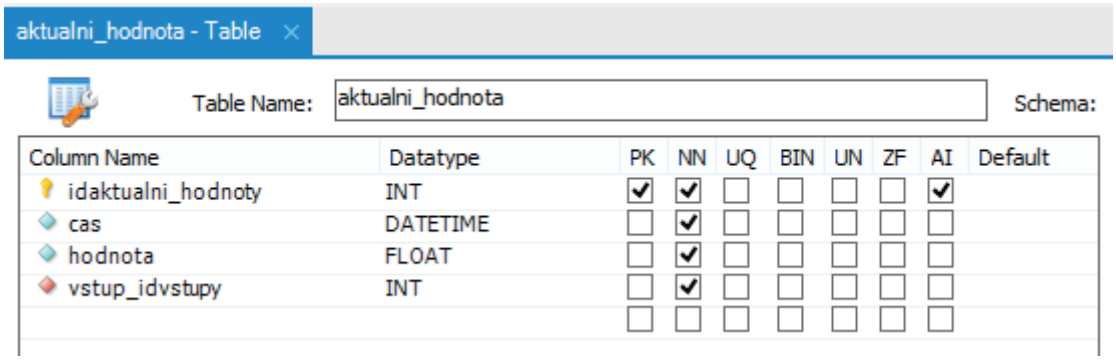



Obrázek 12: MySQL Workbench - atributy entity cas

### 3.3.9 Aktuální hodnota

V entitě aktuální hodnota budou obsažena pouze data týkající se aktuálně naměřených hodnot měřicími přístroji. Tyto hodnoty se nebudou archivovat. Data se v této tabulce budou neustále přepisovat, proto byl zvolen typ úložiště MEMORY. Entita aktuální hodnota by mohla být součástí entity vstupy, ale z důvodu neustálého přepisování hodnot byly atributy týkající se aktuálně naměřených hodnot vyčleněny do zvláštní tabulky, přestože toto vyčlenění nesplňuje normální formy. Vyčlenění bylo provedeno, aby byl možný co nejrychlejší a co nejsnadnější přístup do této entity, aby data v ní obsažená mohla být co nejrychleji přepisována daty aktuálními. Data nebudou jen přepisována ale i čtena a i proto je důležité mít zvolený typ úložiště MEMORY, aby byl přístup ke čtení dat co nejrychlejší.

Tabulka aktuální hodnota bude obsahovat atribut identifikační číslo aktuální hodnoty (idaktualni_hodnota), které je primárním klíčem entity a má nastavené automatické číslování. Dále bude entita obsahovat informace o čase (cas), o aktuální hodnotě naměřené měřicím přístrojem (hodnota), a identifikační číslo vstupu (vstup_idvstup), což je cizí klíč pro provázání s entitou vstupy ve vztahu 1:1. Informace o čase měření není brán z tabulky čas, protože bude přepisován několikrát za minutu a v tabulce čas by se tak hromadila nepotřebná data. Všechny atributy této entity mají nastavená parametr „not null“.

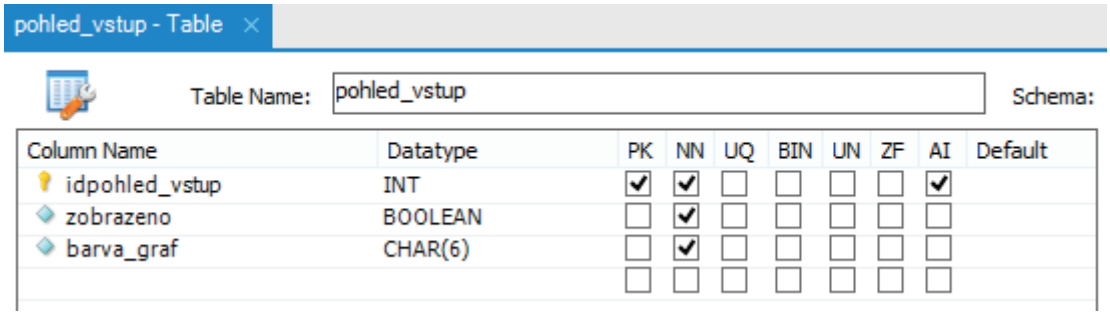



Obrázek 13: MySQL Workbench - atributy entity aktualni_hodnota

### 3.3.10 Pohled vstup

Tabulka pohled_vstup je dekompoziční tabulkou vztahu mezi entitou pohled a vstup, mezi kterými by byl vztah M:N, což porušuje normální formy. Atributy tabulky jsou: identifikační číslo (idpohled_vstup), které je primárním klíčem s nastaveným automatickým číslováním, zobrazeno s datovým typem boolean, který určuje, zda se vybraný vstup bude či nebude v pohledu zobrazovat, a barva grafu. U všech atributů je nastaven parametr „not null“.

Na tuto tabulku jsou navázány entity pohled a vstup. Protože je entita pohled_vstup dekompoziční tabulkou, entita má s těmito entitami vztah N:1, kdy N je na straně entity pohled a entity vstup. S tabulkou pohled_vstup je provázána ještě entita nastavení zobrazení. I mezi těmito entitami je vztah N:1, kde N je na straně entity nastavení zobrazení.



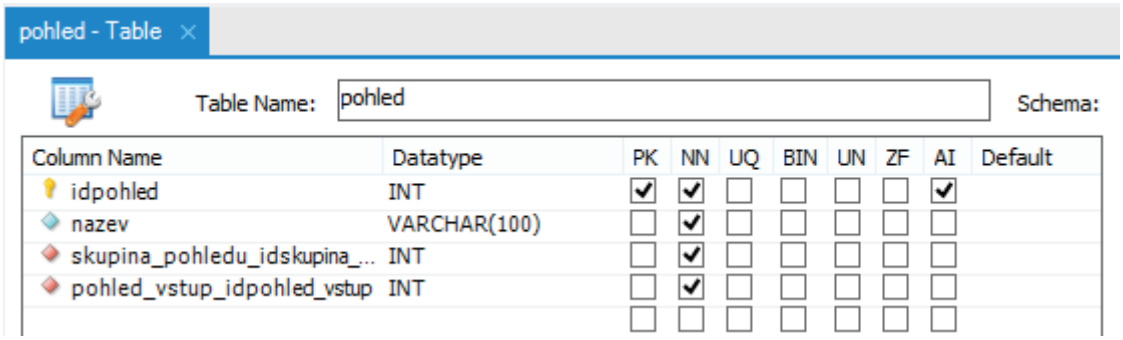

Obrázek 14: MySQL Workbench - atributy pohled_vstup

### 3.3.11 Pohled

Tabulka pohled je propojena s tabulkami nastavení grafu a nastavení zobrazení. S těmito dvěma tabulkami je ve vztahu 1:N. Dále je tabulka pohled provázána s tabulkou skupina

pohledů a s tabulkou pohled_vstup. S těmito entitami má tabulka pohled vztah N:1. Proto entita pohled obsahuje dva cizí klíče zajišťující tyto vazby. Entita pohled_vstup je dekompoziční mezi entitami pohled a vstup, které by byly ve vztahu M:N.

Atribut identifikační číslo pohledu (idpohled) je primárním klíčem této tabulky. Primární klíč má nastavené automatické číslování a parametr „not null“. Mezi další atributy tabulky patří název pohledu (nazev) a dva cizí klíče. První cizí klíč slouží pro vytvoření vazby s dekompoziční tabulkou pohled_vstup (pohled_vstup_idpohled_vstup) a druhý cizí klíč slouží k provázání tabulky pohled s tabulkou skupina pohledů (skupina_pohledu_idskupina_pohledu). Všechny atributy entity pohled mají nastavený parametr „not null“.

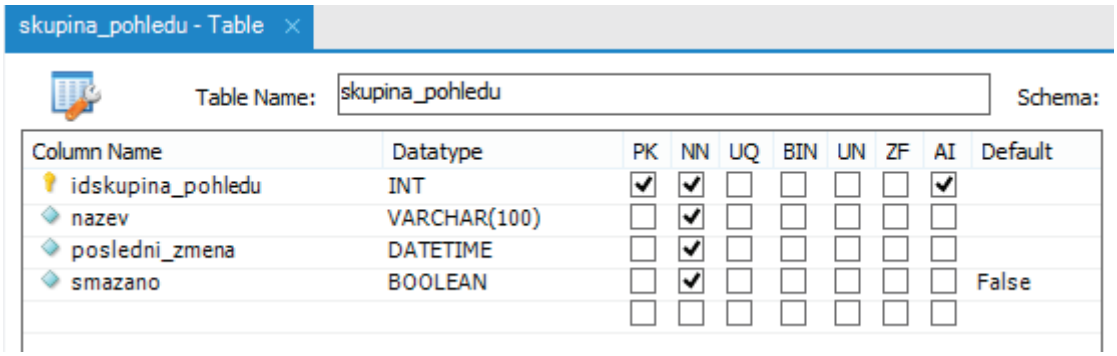



Obrázek 15: MySQL Workbench - atributy entity pohled

### 3.3.12 Skupina pohledů

Pomocí této entity se seskupuje několik pohledů do jedné skupiny. Pro jednoznačnou identifikaci byl použit jako primární klíč atribut identifikační číslo skupiny pohledů (idskupina_pohledu) s nastaveným automatickým číslováním. Tabulka skupiny pohledů je navázána pouze na tabulku pohledů a mají mezi sebou vztah 1:N, kdy N je na straně tabulky pohled.

Dalšími atributy entity jsou název skupiny pohledů (nazev), poslední změna (posledni_zmena), což je datum a čas poslední změny dat v daném řádku tabulky, tento atribut má datový typ datetime. Posledním atributem je smazáno (smazano), který uchovává informaci, zda daná skupina pohledů nebyla už někdy vytvořena a odstraněna. Výchozí hodnota tohoto atributu je nastavena na „false“. Atribut smazáno je datového typu boolean. Pomocí atributu smazáno bude odstraněný pohled možno z databáze rychle a jednoduše obnovit.



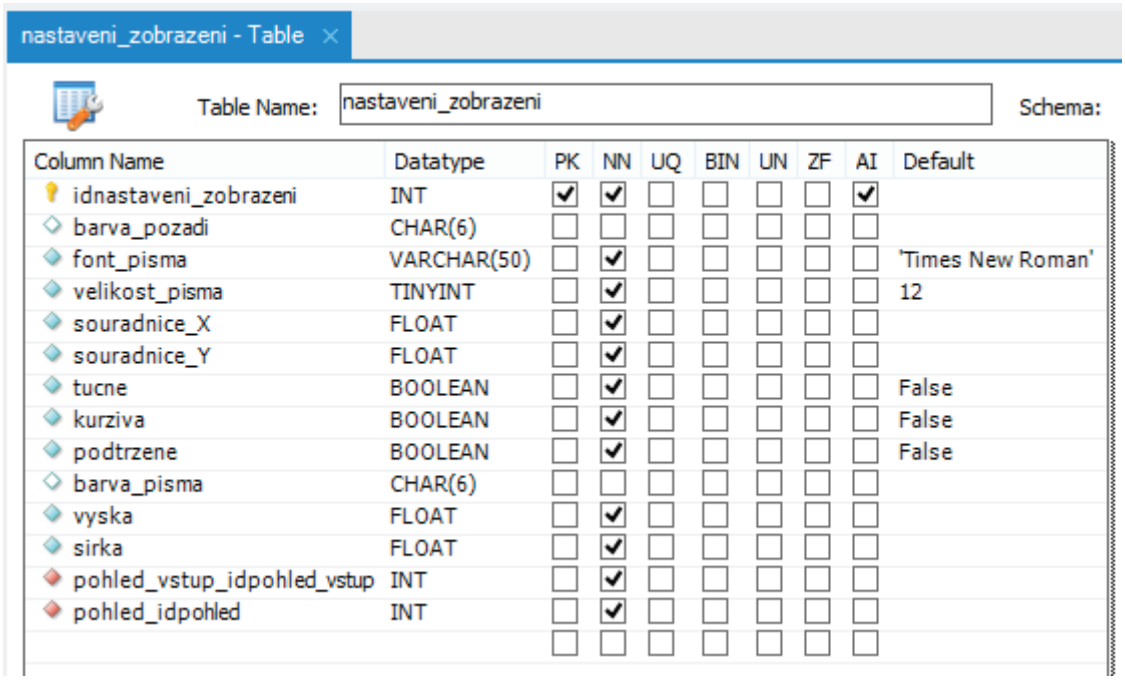

Obrázek 16: MySQL Workbench - atributy entity skupina_pohledu

### 3.3.13 Nastavení zobrazení

Tato entita má za úkol uchovávat veškeré informace o nastavení zobrazení pohledu jako celku a jednotlivých vstupů v pohledu zobrazených. Proto je entita nastavení zobrazení ve vztahu s tabulkou pohled a tabulkou pohled_vstup. Oba vztahy entit jsou N:1, kdy N je na straně tabulky nastavení zobrazení.

Tabulka má celkem čtrnáct atributů, z toho dva jsou cizí klíče zajišťující výše uvedené vazby na entity pohled a pohled_vstup. Primárním klíčem tabulky je identifikační číslo (idnastaveni_zobrazeni). Primární klíč má nastavený parametr „not null“ a automatické číslování. Dalšími atributy slouží k uložení nastavení, patří sem: nastavení barvy pozadí (barva_pozadi), nastavení písma – ať už fontu (font_pisma), velikosti (velikost_pisma), barvy (barva_pisma) či nastavení zda má být písmo psané kurzívou, tučně, nebo zda má být podržené (kurziva., tucne, podtrzene), velikosti pohledu či vstupu v pohledu, kdy velikost je určena výškou a šířkou vyhrazeného místa pro pohled nebo vstup v pohledu zobrazený. Poslední dva atributy uchovávají informace o umístění pohledu na obrazovce pomocí souřadnic osy X a Y (souradnice_X, souradnice_Y).

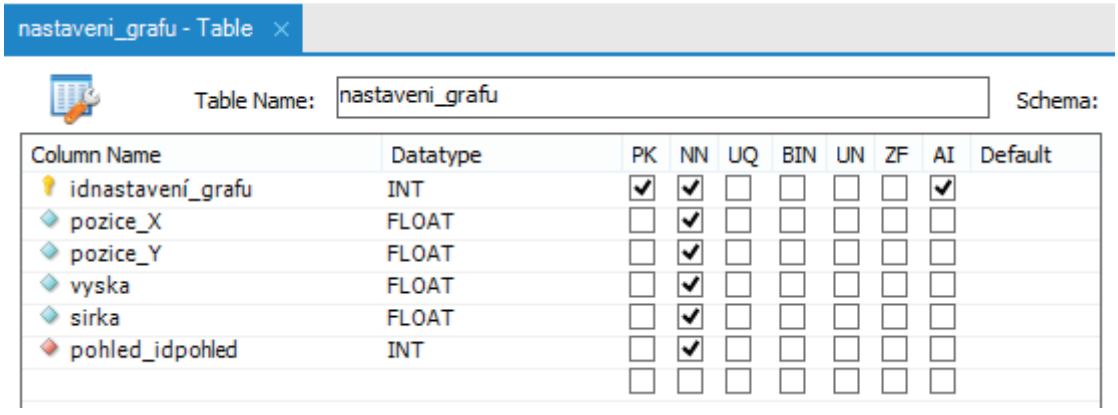
Většina atributů má nastavený parametr „not null“, jen atributy barva pozadí a barva písma tento parametr nastavený nemají.




Obrázek 17: MySQL Workbench - atributy entity nastaveni_zobrazeni

### 3.3.14 Nastavení grafu

Tato entita uchovává veškeré informace o nastavení grafu, které je určeno ovládací aplikací. Entita obsahuje celkem 8 atributů, z toho jeden atribut je identifikační číslo nastavení grafu (idnastaveni_grafu), další je cizí klíč vytvářející vazbu s entitou pohled (pohled_idpohled). Další atributy již obsahují informace o nastavení grafu jako je nastavení pozice zobrazení na monitoru zobrazovacího zařízení (pozice_X, pozice_Y) a informace o velikosti zobrazovaného grafu na monitoru zobrazovacího zařízení (vyska, sirka). Všechny atributy mají nastavený parametr „not null“a primární klíč má nastavené, tak jako ostatní primární klíče v tomto návrhu databáze automatické číslování.




Obrázek 18: MySQL Workbench - atributy entity nastaveni_grafu

### 3.3.15 Shrnutí fáze logického návrhu

Ve fázi logického návrhu byl převeden konceptuální návrh množiny relačních tabulek, která byla zpracována i graficky (obrázek 4). Dále byly v této fázi rozpracovány jednotlivé entity – byly nadefinovány jejich primární klíče, jednotlivé potřebné atributy a cizí klíče k uskutečnění navrhovaných vztahů z konceptuálního návrhu. Normální formy byly porušeny pouze v případě vztahu entity vstup s entitou aktualni_hodnota. Toto porušení normálních forem je odůvodněno potřebou rychlého přístupu do tabulky aktualni_hodnota jak z důvodu čtení, tak z důvodu zápisu – v této tabulce nebudou data uchovávána, ale přepisována.

## 3.4 Fáze fyzického návrhu

Fyzický návrh databáze navazuje na logický návrh databáze. Ve fyzickém návrhu databáze je rozhodováno o procesu implementace. Fyzický návrh má popisovat podkladové tabulky, organizaci souborů, použité indexy, integritní omezení a bezpečnostní omezení. Aby mohl být realizován fyzický návrh, návrhář musí znát počítačový systém a funkčnost cílového databázového systému.

Společnost ELTEX electronics si bude fázi fyzického návrhu a implementaci zajišťovat sama, proto se předkládaná práce fází fyzického návrhu databáze dále nezabývá.

# 4 ZÁVĚR

Cílem této bakalářské práce bylo navrhnout databázové řešení přizpůsobeného potřebám a požadavkům společnosti ELTEX electronics. Databázové řešení je určené pro archivaci a následnou vizualizaci dat získaných z měřicích přístrojů. Vizualizace bude prováděna pomocí aplikace, kterou si společnost navrhuje samostatně. Navrhované databázové řešení bylo zpracováváno pro MySQL server. Při zpracování bylo využito programu MySQL Workbench verze 6.0. Tento program byl využit především pro možnost snadného vizuálního zpracování návrhu struktury databáze, ale i pro jeho intuitivní ovládání. Je vydáván pod licencí General Public Licence a jedná se tedy o open source software.

Navrhovaná databáze by měla být implementována nejen na databázový server, ale i na jednotlivé měřící počítačové stanice, aby bylo umožněno ukládání dat i při výpadku komunikace se serverem ať už způsobeným výpadkem serveru, nebo výpadkem komunikace mezi serverem a měřicími stanicemi.

Při zpracovávání návrhu databáze bylo postupováno podle konceptuální metodologie se zapracováním požadavků společnosti ELTEX electronics, který je detailněji rozepsán v praktické části této práce. Během zpracovávání práce byl návrh databáze průběžně konzultován se společností, aby co nejlépe vyhovoval jejich potřebám a požadavkům. Zvolená konceptuální metodologie má tři fáze návrhu: konceptuální fázi, logickou fázi a fyzickou fázi. V práci je zpracována pouze fáze konceptuální a logická, protože fyzickou fázi návrhu si společnost bude zpracovávat sama. Ve fyzické fázi návrhu je rozhodováno o procesu implementace a ta již není součástí této práce. I zpracování aplikace s uživatelským prostředím bude provedeno samostatně společností ELTEX electronics.

Tato bakalářská práce naplnila svůj cíl a navržená databáze by měla zjednodušit a zefektivnit archivaci a následnou vizualizaci dat.

V příloze předkládané práce je i skript na vytvoření zpracovaného návrhu databáze pro MySQL server.

# SEZNAM POUŽITÉ LITERATURY

[1] JUNEXT. Český manuál k relačnímu databázovému systému MySQL. *Juntex.cz* [online]. © 1995-2013 Oracle Corporation and/or its affiliates, © 2002-2013 Junext [cit. 2013-11-17]. Dostupné z: http://www.junext.net/mysql/

[2] ORACLE. Getting Started with MySQL. Dev.mysql.com [online]. © 2013, Oracle Corporation and/or its affiliates [cit. 2013-11-16]. Dostupné z: http://dev.mysql.com/tech-resources/articles/mysql_intro.html

[3] SCHNEIDER, Robert D. *MySQL Oficiální průvodce tvorbou, správou a laděním databází*. GRADA Publishing, 2006. 372 s. ISBN 80-247-1516-3. Dostupné také z: http://books.google.cz

[4] HODAN, Petr. Seznámení s SQL. *Root.cz* [online]. 2000 [cit. 2013-11-15]. ISSN 212-8309. Dostupné z: http://www.root.cz/clanky/uvod-do-sql/#ic=serial-box&icc=text-title

[5] PAETA, Petr. *Co programátory ve škole neučí aneb Softwarové inženýrství v reálné praxi*. Brno: Computer Press, 2003. ISBN 80-251-0073-1.

[6] CVRČEK, Pavel. Začínáme s MySQL 1. Díl. *Živě.cz* [online]. ©2013 Mladá fronta a.s. [cit. 2013-11-18]. Dostupné z: http://www.zive.cz/clanky/zaciname-s-mysql-1dil/sc-3-a-102589/default.aspx

[7] ŠIMKO, Martin a Zdeněk LEHOCKÝ. Přehled datových typů v MySQL. *Programujte.com* [online]. 2007. Roč. 2007, č. 7 [cit. 2013-11-24]. ISSN 1801-1586. Dostupné z: http://programujte.com/clanek/2007052903-prehled-datovych-typu-v-mysql/

[8] ZAJÍC, Petr. MySQL (31) – Indexy. *Linuxsoft.cz* [online]. 2005 [cit. 2014-01-24]. ISSN 1801-3805. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=912

[9] BEGG, C., R. HOLOWCZAK, a T. CONOLLY. *Mistrovství databáze: Profesionální průvodce tvorbou efektivních databází*. Praha: Computer Press, 2009. 584 s. ISBN 9788025123287.

[10] BRUCKNER, T., VOŘÍŠEK, J., BUCHALCEVOVÁ, A., STANOVSKÁ, I., CHLAPEK, D. a ŘEPA, V. *Tvorba informačních systémů: principy, metodiky, architektury*. Praha: Grada Publishing, 2012. 347 s. ISBN 978-80-247-4153-6.

[11] KOCH, Miloš. *Datové a funkční modelování.* Brno: Akademické nakladatelství CERM, 2004. 108 s. ISBN 80-214-2724-8.

# SEZNAM TABULEK

Tabulka 1: Celočíselné typy (Zdroj [1]) ..... 18

Tabulka 2: Číselné typy s pohyblivou desetinnou čárkou (Zdroj [1]) ..... 18

Tabulka 3: Řetězcové typy (Zdroj [7]) ..... 19

Tabulka 4:Typy pro datum a čas (Zdroj [1]) ..... 20

# SEZNAM OBRÁZKŮ

Obrázek 1: relační datová struktura (Zdroj [9]) ..... 22

Obrázek 2: Schéma síťové infrastruktury ..... 34

Obrázek 3: MySQL Workbench - nová tabulka ..... 36

Obrázek 4: MySQL Workbench - diagram entit a vztahů mezi nimi ..... 40

Obrázek 5: MySQL Workbench - atributy entity merici_stanice ..... 41

Obrázek 6: MySQL Workbench - atributy entity merici_pristroj ..... 41

Obrázek 7: MySQL Workbench - atributy entity typ_pristroje ..... 42

Obrázek 8: MySQL Workbench - atributy entity vstup ..... 42

Obrázek 9: MySQL Workbench - atributy entity jednotka ..... 43

Obrázek 10: MySQL Workbench - atributy entity rozsah ..... 44

Obrázek 11: MySQL Workbench - atributy entity hodnota ..... 44

Obrázek 12: MySQL Workbench - atributy entity cas ..... 45

Obrázek 13: MySQL Workbench - atributy entity aktualni_hodnota ..... 46

Obrázek 14: MySQL Workbench - atributy pohled_vstup ..... 46

Obrázek 15: MySQL Workbench - atributy entity pohled ..... 47

Obrázek 16: MySQL Workbench - atributy entity skupina_pohledu ..... 48

Obrázek 17: MySQL Workbench - atributy entity nastaveni_zobrazeni ..... 49

Obrázek 18: MySQL Workbench - atributy entity nastaveni_grafu ..... 49

# SEZNAM PŘÍLOH

PŘÍLOHA Č. 1 – schéma síťové infrastruktury ..... I

PŘÍLOHA Č. 2 – diagram navrhované databáze ..... II

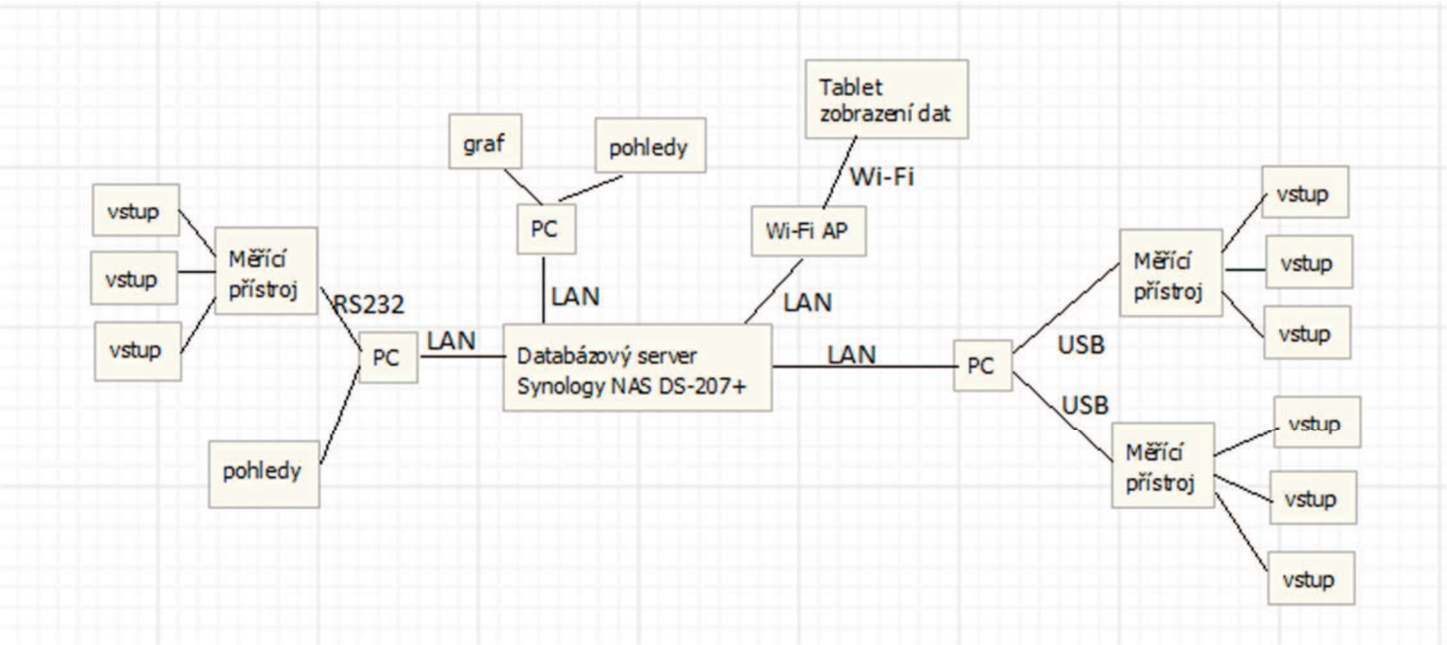
PŘÍLOHA Č. 3 – číselné datové typy ..... III

PŘÍLOHA Č. 4 – datové typy pro datum a čas ..... IV

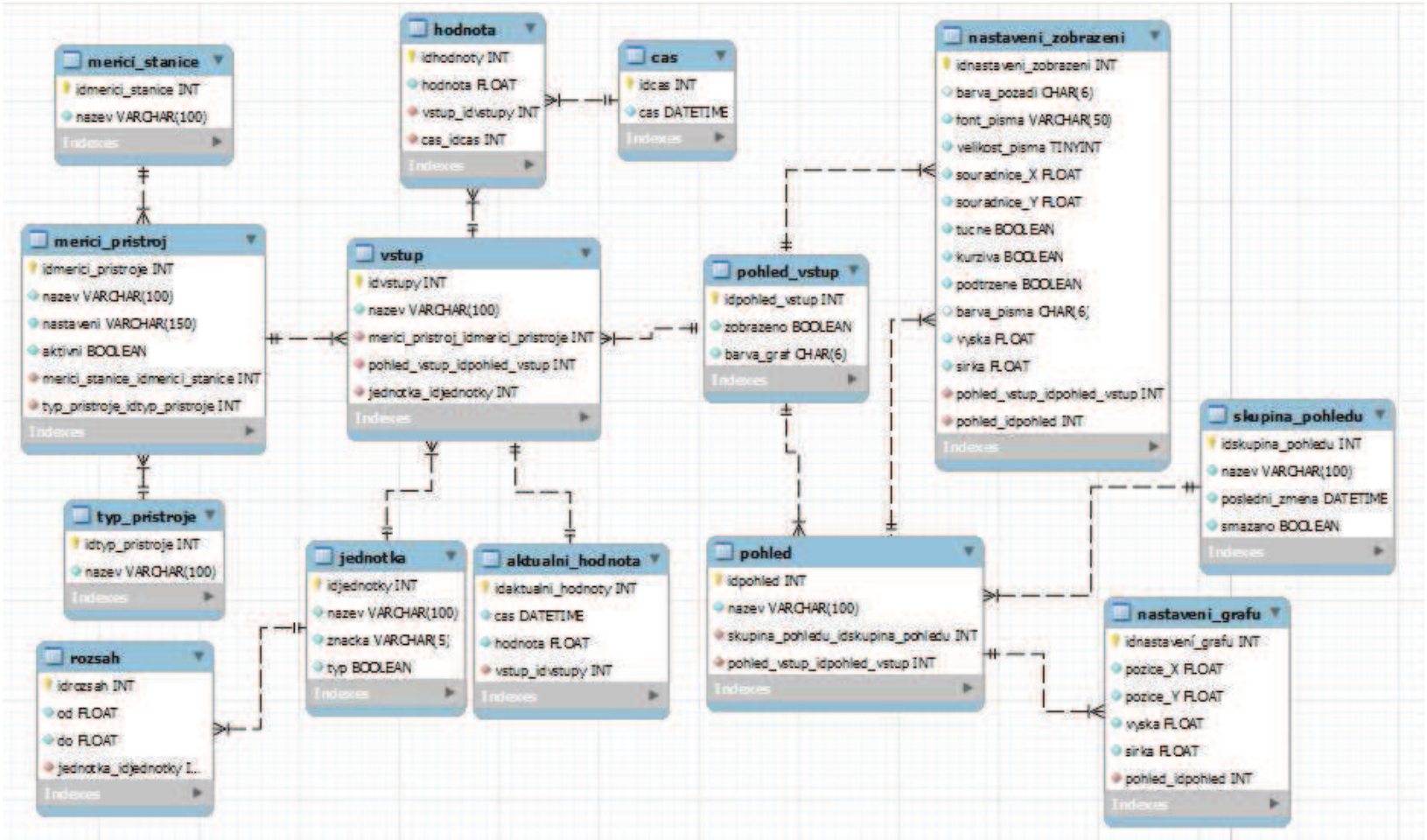
PŘÍLOHA Č. 5 – řetězcové datové typy ..... V

PŘÍLOHA Č. 6 – SQL skript ..... VI

**PŘÍLOHA Č. 1 – schéma síťové infrastruktury**



**PŘÍLOHA Č. 2 – diagram navrhované databáze**



**PŘÍLOHA Č. 3 – číselné datové typy**

| Datový typ | Rozsah | | Velikost | Výchozí hodnota |
|---|---|---|---|---|
| | se znaménkem | bez znaménka | | |
| **TYNYINT** | -128 až 127 | 0 až 255 | 1 bajt | Null / 0 |
| **SMALLINT** | -32768 až 32767 | 0 až 65535 | 2 bajty | Null / 0 |
| **MEDIUMINT** | -8388608 až 8388607 | 0 až 16777215 | 3 bajty | Null / 0 |
| **INT** | -2147483648 až 2147483647 | 0 až 4294967295 | 4bajty | Null / 0 |
| **FLOAT** | ±1,175494351E−38 až ±3,402823466E+38 | | 4 bajty | Null / 0 |
| **DOUBLE** | ±2,2250738585072014E-308 až ±1,17976931348623157E+30 | | 8 bajtů | Null / 0 |
| **DECIMAL** | ±2,2250738585072014E-308 až ±1,17976931348623157E+30 | | pohyblivé | Null / 0 |

**PŘÍLOHA Č. 4 – datové typy pro datum a čas**

| Datový typ | Formát | Rozsah | Velikost |
|---|---|---|---|
| **DATE** | 'RRRR-MM-DD' | '1000-01-01' až '9999-12-31' | 3 bajty |
| **DATETIME** | 'RRRR-MM-DD hh:mm:ss' | '1000-01-01 00:00:00' až '9999-12-31 23:59:59' | 8 bajtů |
| **TIMESTAMP** | 'RRRR-MM-DD hh:mm:ss' | '1970-01-01 00:00:00' až '2037-01-01 00:00:00' | 4 bajty |
| **TIME** | 'hh:mm:ss' a '-hh:mm:ss' | '-838:59:59' až '838:59:59' | 3 bajty |
| **YEAR (4)** | RRRR | '1901' až '2155' a '0000' | 1 bajt |
| **YEAR (2)** | RR | '1970' až '2069' | 1 bajt |

**PŘÍLOHA Č. 5 – řetězcové datové typy**

| Datový typ | Povolená délka | Velikost |
|---|---|---|
| **CHAR** | 0 až 255 bajtů | až 255 bajtů |
| **VARCHAR** | 0 až 255 bajtů | hodnota délky + 1 bajt |
| **TINYTEXT** | 0 až 255 bajtů | hodnota délky + 1 bajt |
| **TEXT** | 0 až 65535 bajtů | hodnota délky + 2 bajty |
| **MEDIUMTEXT** | 0 až 16777245 bajtů | hodnota délky + 3 bajty |
| **LONGTEXT** | 0 až 4294967295 bajtů | hodnota délky + 4 bajty |

## PŘÍLOHA Č. 6 – SQL skript

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `BP_navrh_databaze` DEFAULT CHARACTER SET
utf8 COLLATE utf8_general_ci ;
USE `BP_navrh_databaze` ;

-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`merici_stanice`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`merici_stanice` (
  `idmerici_stanice` INT NOT NULL AUTO_INCREMENT,
  `nazev` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idmerici_stanice`))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`typ_pristroje`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`typ_pristroje` (
  `idtyp_pristroje` INT NOT NULL AUTO_INCREMENT,
  `nazev` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idtyp_pristroje`))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`merici_pristroj`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`merici_pristroj` (
  `idmerici_pristroje` INT NOT NULL AUTO_INCREMENT,
  `nazev` VARCHAR(100) NOT NULL,
  `nastaveni` VARCHAR(150) NOT NULL,
  `aktivni` TINYINT(1) NOT NULL DEFAULT True,
  `merici_stanice_idmerici_stanice` INT NOT NULL,
  `typ_pristroje_idtyp_pristroje` INT NOT NULL,
  PRIMARY KEY (`idmerici_pristroje`),
  INDEX `fk_merici_pristroj_merici_stanice_idx`
(`merici_stanice_idmerici_stanice` ASC),
  INDEX `fk_merici_pristroj_typ_pristroje1_idx`
(`typ_pristroje_idtyp_pristroje` ASC))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`pohled_vstup`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`pohled_vstup` (
  `idpohled_vstup` INT NOT NULL AUTO_INCREMENT,
  `zobrazeno` TINYINT(1) NOT NULL,
  `barva_graf` CHAR(6) NOT NULL,
  PRIMARY KEY (`idpohled_vstup`))
ENGINE = MyISAM;
```

```
-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`jednotka`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`jednotka` (
  `idjednotky` INT NOT NULL AUTO_INCREMENT,
  `nazev` VARCHAR(100) NOT NULL,
  `znacka` VARCHAR(5) NOT NULL,
  `typ` TINYINT(1) NOT NULL,
  PRIMARY KEY (`idjednotky`))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`vstup`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`vstup` (
  `idvstupy` INT NOT NULL AUTO_INCREMENT,
  `nazev` VARCHAR(100) NOT NULL,
  `merici_pristroj_idmerici_pristroje` INT NOT NULL,
  `pohled_vstup_idpohled_vstup` INT NOT NULL,
  `jednotka_idjednotky` INT NOT NULL,
  PRIMARY KEY (`idvstupy`),
  INDEX `fk_vstup_merici_pristroj1_idx`
(`merici_pristroj_idmerici_pristroje` ASC),
  INDEX `fk_vstup_pohled_vstup1_idx` (`pohled_vstup_idpohled_vstup`
ASC),
  INDEX `fk_vstup_jednotka1_idx` (`jednotka_idjednotky` ASC))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`cas`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`cas` (
  `idcas` INT NOT NULL AUTO_INCREMENT,
  `cas` DATETIME NOT NULL,
  PRIMARY KEY (`idcas`))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`hodnota`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`hodnota` (
  `idhodnoty` INT NOT NULL AUTO_INCREMENT,
  `hodnota` FLOAT NOT NULL,
  `vstup_idvstupy` INT NOT NULL,
  `cas_idcas` INT NOT NULL,
  PRIMARY KEY (`idhodnoty`),
  INDEX `fk_hodnota_vstup1_idx` (`vstup_idvstupy` ASC),
  INDEX `fk_hodnota_cas1_idx` (`cas_idcas` ASC))
ENGINE = MyISAM;




-- -----------------------------------------------------
```

```
-- Table `BP_navrh_databaze`.`rozsah`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`rozsah` (
  `idrozsah` INT NOT NULL AUTO_INCREMENT,
  `od` FLOAT NOT NULL,
  `do` FLOAT NOT NULL,
  `jednotka_idjednotky` INT NOT NULL,
  PRIMARY KEY (`idrozsah`),
  INDEX `fk_rozsah_jednotka1_idx` (`jednotka_idjednotky` ASC))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`aktualni_hodnota`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`aktualni_hodnota` (
  `idaktualni_hodnoty` INT NOT NULL AUTO_INCREMENT,
  `cas` DATETIME NOT NULL,
  `hodnota` FLOAT NOT NULL,
  `vstup_idvstupy` INT NOT NULL,
  PRIMARY KEY (`idaktualni_hodnoty`),
  INDEX `fk_aktualni_hodnota_vstup1_idx` (`vstup_idvstupy` ASC))
ENGINE = MEMORY;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`skupina_pohledu`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`skupina_pohledu` (
  `idskupina_pohledu` INT NOT NULL AUTO_INCREMENT,
  `nazev` VARCHAR(100) NOT NULL,
  `posledni_zmena` DATETIME NOT NULL,
  `smazano` TINYINT(1) NOT NULL DEFAULT False,
  PRIMARY KEY (`idskupina_pohledu`))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`pohled`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`pohled` (
  `idpohled` INT NOT NULL AUTO_INCREMENT,
  `nazev` VARCHAR(100) NOT NULL,
  `skupina_pohledu_idskupina_pohledu` INT NOT NULL,
  `pohled_vstup_idpohled_vstup` INT NOT NULL,
  PRIMARY KEY (`idpohled`),
  INDEX `fk_pohled_skupina_pohledu1_idx`
(`skupina_pohledu_idskupina_pohledu` ASC),
  INDEX `fk_pohled_pohled_vstup1_idx` (`pohled_vstup_idpohled_vstup`
ASC))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`nastaveni_grafu`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`nastaveni_grafu` (
  `idnastavení_grafu` INT NOT NULL AUTO_INCREMENT,
  `pozice_X` FLOAT NOT NULL,
  `pozice_Y` FLOAT NOT NULL,
```

```
  `vyska` FLOAT NOT NULL,
  `sirka` FLOAT NOT NULL,
  `pohled_idpohled` INT NOT NULL,
  PRIMARY KEY (`idnastavení_grafu`),
  INDEX `fk_nastaveni_grafu_pohled1_idx` (`pohled_idpohled` ASC))
ENGINE = MyISAM;


-- -----------------------------------------------------
-- Table `BP_navrh_databaze`.`nastaveni_zobrazeni`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BP_navrh_databaze`.`nastaveni_zobrazeni` (
  `idnastaveni_zobrazeni` INT NOT NULL AUTO_INCREMENT,
  `barva_pozadi` CHAR(6) NULL,
  `font_pisma` VARCHAR(50) NOT NULL DEFAULT 'Times New Roman',
  `velikost_pisma` TINYINT NOT NULL DEFAULT 12,
  `souradnice_X` FLOAT NOT NULL,
  `souradnice_Y` FLOAT NOT NULL,
  `tucne` TINYINT(1) NOT NULL DEFAULT False,
  `kurziva` TINYINT(1) NOT NULL DEFAULT False,
  `podtrzene` TINYINT(1) NOT NULL DEFAULT False,
  `barva_pisma` CHAR(6) NULL,
  `vyska` FLOAT NOT NULL,
  `sirka` FLOAT NOT NULL,
  `pohled_vstup_idpohled_vstup` INT NOT NULL,
  `pohled_idpohled` INT NOT NULL,
  PRIMARY KEY (`idnastaveni_zobrazeni`),
  INDEX `fk_nastaveni_zobrazeni_pohled_vstup1_idx`
(`pohled_vstup_idpohled_vstup` ASC),
  INDEX `fk_nastaveni_zobrazeni_pohled1_idx` (`pohled_idpohled` ASC))
ENGINE = MyISAM;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```