Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Vyšší odborná škola informačních služeb v Praze

Ladislav Šipka

Analýza a návrh informačního systému pro IT infrastrukturu VOŠIS

Bakalářská práce

2008

Prohlášení

Prohlašuji, že jsem bakalářskou práci na téma
„Analýza a návrh informačního systému pro IT infrastrukturu VOŠIS“
zpracoval samostatně a použil pouze zdrojů, které cituji a uvádím v seznamu použité literatury.

V Praze dne 02.01.2008 *Podpis*

Obsah:

Abstrakt

Cílem a hlavní otázkou práce bylo najít optimální řešení problému analýzy a návrhu infrastrukturní databáze pro potřeby Vyšší odborné školy informačních služeb v Praze.
S využitím teoretických i praktických znalostí, získaných během studia, povinné školní praxe i několikaleté práce v oboru jsem se zaměřil na použitelnost a univerzálnost mnou navržené infrastrukturní databáze.

Práce je dělena do několika kapitol. Úvodní kapitola je zaměřena na objasnění důvodů, proč jsem si zvolil toto téma, na analýzu současného stavu evidence infrastruktury na Vyšší odborné škole informačních služeb v Praze, na analýzu potřeby evidence infrastruktury a na definici zadání a požadavků ze strany Vyšší odborné školy informačních služeb.

Následující kapitola obsahuje rozbor některých řešení problému, jejich popis, klady, zápory a vzájemné porovnání.

Další kapitoly již obsahují jedno zvolené řešení, jeho podrobnou analýzu, popis, objasnění otázky proč bylo vybráno právě toto řešení za nejvhodnější a nástin dalších možných obměn tohoto řešení infrastrukturní databáze v závislosti na změně zadání.

V závěru práce poukazuji na přínosy plynoucí z této bakalářské práce.

1. Úvod

Jako téma své Bakalářské práce jsem zvolil řešení problému analýzy a návrhu infrastrukturní databáze pro potřeby Vyšší odborné školy informačních služeb v Praze, neboť věřím, že se jedná o zajímavé téma, které se svou komplexností dotýká mnoha oblastí IT. Počínaje analýzou, přes metodiku a návrh procházeje a samotným vytvořením databáze konče.

K vyřešení je možné přistupovat několika různými způsoby, můžeme se pohybovat pouze na teoretické úrovni hypotéz, pouček a pravidel pro správný návrh databází, nebo se můžeme pohybovat na úrovni samotné analýzy, či právě naopak se může věnovat více prostoru samotnému návrhu databáze. Samotný návrh databáze lze také realizovat rozdílně. Je možné vytvořit uživatelsky přívětivou databázi, která ovšem nesplňuje všechna pravidla pro normování databáze, či můžeme vytvořit databázi, která bezvýhradně odpovídá všem pravidlům pro normování, avšak částečně na úkor uživatelské přívětivosti.

Ve své práci se nesnažím najít odpověď na otázku, zda je lepší normovaná, či uživatelsky přívětivá databáze. Nesnažím se najít ideální a dokonalou databázi. Snažím se pouze s využitím svých praktických zkušeností z oboru najít ideální poměr, který úspěšně povede ke splnění primárního cíle a tím je vytvoření infrastrukturní databáze pro potřeby Vyšší odborné školy informačních služeb v Praze odpovídající zadání.

Pokud budu v následujícím textu psát o infrastruktuře, mám tím na mysli infrastrukturu informačního systému, nikoli například dopravní infrastrukturu.

1.1. Zadání Bc. práce

Hlavním cílem mé bakalářské práce je vytvořit databázi pro evidenci infrastruktury Vyšší odborné školy informačních služeb v Praze na základě zadání požadavků ze strany školy.

Požadavky na databázi evidence infrastruktury byly stanoveny takto:

I. Databáze by měla být primárně určena pro evidenci infrastruktury.

II. Databáze by měla být použitelná pro evidenci dalšího vybavení, které není součásti počítačového vybavení ani s ním nijak nesouvisí.

III. Databáze by měla být připravena i na evidenci majetku pro ekonomické a účetní potřeby.

IV. Databáze by měla být snadno udržovatelná z hlediska vkládání, získávání, updatu a mazání dat.

V. Databáze by měla být snadno rozšířitelná bez větších zásahů a nutnosti měnit strukturu databáze nebo dat.

VI. Databáze by měla být založena na volně dostupné databázové platformě.

VII. Pro práci s databází bude používáno rozhraní napsané v jazyce PHP.

1.2. Současný stav evidence infrastruktury a majetku

Potřeba ucelené a přehledné evidence infrastruktury vychází z potřeby přehledu infrastrukturních prvků. Proč je to ale důležité?

Důležitost evidence spočívá v možnosti poskytování ucelených přehledů o nákladech, počtu, výkonu a rozmístění jednotlivých prvků. Napomáhá dokumentovat složité vazby mezi jednotlivými objekty a subjekty, které jsou součástí infrastruktury, zasahují do ní, či s ní jinak manipulují. Poskytuje informace napomáhající při rozhodování o budoucím uspořádání infrastruktury a do jisté míry ovlivňuje rozhodování i o okolních systémech.

K evidenci infrastruktury a majetku lze přistupovat mnoha způsoby, je možné obojí evidovat v jedné databázi, čí dokonce v jedné jediné tabulce, lze evidovat zvlášť infrastrukturu a zvlášť ostatní majetek, lze evidovat vše v jednom jediném systému, či v několika rozdílných systémech. Evidence může být v rámci jedné databáze, nebo v rámci několika různých databází. Každý ze způsobů evidence přináší své výhody a samozřejmě i své nevýhody.

V současné době je evidence infrastruktury a majetku na Vyšší odborné škole informačních služeb v Praze vedena podle umístění v místnostech, kde ke každé místnosti patří určitá skupina zařízení. Infrastruktura i ostatní vybavení je evidováno pohromadě bez výraznějšího odlišení podle jednotlivých typů majetku. Jak je zřejmé, vyhledávání v takovéto evidenci jistě není zcela ideální a to jak z pohledu rychlosti, jednoduchosti tak i validity výsledků. Takovýto systém evidence je poměrně obtížné udržovat a upravovat. Vzhledem k tomu, že vše je udržováno v určitém dokumentu/dokumentech, je také obtížné řídit přístupy k tomuto souboru v případě, že s ním musí pracovat více lidí najednou.

Převedením tohoto způsobu evidence do podoby databáze eliminujeme většinu negativních dopadů včetně přístupů uživatelů a jejich oprávnění. Celý systém se zpřehlední, usnadní se hledání i administrace a pomocí skriptů je možné provádět pravidelnou zálohu databáze nebo pravidelné reportování změn.

1.3. Proč databázi?

Proč je výhodné použít právě databázi? Oproti současnému stavu evidence má použití databáze hned několik výhod.

- Databáze poskytuje přehlednější pohled na evidovaná zařízení.
- Data mohou být uložena na jednom místě, čímž se organizace dat stává přehlednější a lépe organizovanou.
- Kromě textových hodnot lze do databáze vkládat i vektorovou a rastrovou grafiku a další typy souborů.
- Organizace dat do databází umožňuje snazší vyhledávání, vkládání, update i mazání dat.
- Díky použití databáze je možné efektivněji řídit přístupy k datům a práva uživatelů.
- Databáze přináší vyšší bezpečnost dat.
- Použitím databází lze vytvořit i poměrně složité znalostní báze, archivy požadavků, činností a incidentů, evidovat historii nebo logovat změny v samotné databázi.
- Spojením databáze a patřičných skriptů, například PHP, je možné provádět snadné importy nebo naopak exporty dat a výsledků hledání.

Bohužel použití databáze nemá pouze výhody, ale najde se i pár nevýhod.

- Administrace databáze i práce s ní vyžaduje alespoň elementární znalosti dotazovacího jazyka SQL a principů fungování databází.
- Databáze pro svůj běh potřebuje počítač, nebo server s příslušným softwarem.
- Většina databázových platforem není k dispozici zdarma, nebo s nějakými omezeními.
- Nedostatek kvalitních, volně dostupných databázových platforem.

2. Možné přístupy k řešení zadání

Stejně jako lze k hledání řešení přistupovat různě, lze nalézt i různá řešení. V této části bych rád uvedl některé z možných přístupů vedoucích k úspěšnému splnění zadání, pár osobních postřehů a faktů, kterých je vhodné si všimnout a které mohou zásadním způsobem ovlivnit rozhodování při tvorbě a návrhu databáze. Stručně zde představím pár možných návrhů a provedu jejich porovnání.

2.1. Rozbor zadání

Předpokládejme, že výstupem této práce má být databáze pro evidenci infrastrukturních prvků, mapování infrastruktury a evidenci ostatního majetku, splňující požadavky ze strany zadavatele. Samozřejmostí, než začneme navrhovat samotnou databázi, je nutnost rozebrat podrobně požadavky, provést analýzu, vytvořit první návrhy a přesvědčit se, že odpovídají zadání.

Vzhledem k tomu že primární účel námi navržené databáze je evidence infrastruktury, měli by naše kroky při návrhu směřovat k cíli, jímž je co nejpřesnější popis reality a současného stavu infrastruktury. Jen tak totiž budeme schopni odhalit slabá místa a předejít problémům či výpadkům. Ze zadání dále vyplívá, že naše databáze nebude sloužit pouze pro evidenci infrastruktury, ale i jako evidence veškerého majetku pro účetní a inventární potřeby.

Vzhledem k požadavku týkajícího se databázové platformy jsme nuceni použít některé z volně dostupných a použitelných databázových platforem jako jsou MySQL[1], Firebird[2] či PostgreSQL[3]. Přestože jsou tyto databázové systémy v mnohém srovnatelné s velkými hráči na trhu, stále bohužel nedosahují kvalit svých komerčních konkurentů, jako jsou Oracle, Microsoft SQL, IBM DB2 a jiné. Pro naše účely jsou ovšem volně dostupné databázové systémy postačující a vzhledem k předpokládané velikosti a rozsahu DB by bylo zbytečné pořizovat komplexní databázová řešení v podobě velkých softwarových balíků.

[1] MySQL je registrovaná ochranná známka společnosti MySQL AB. Databázový systém MySQL je šířen pod licencí GPL, nebo pod komerční licencí, viz.[http://www.mysql.com/]

[2] Firebird, nebo FirebirdSQL jsou ochranné známky organizace Firebird Foundation inc. Databázový systém Firebird je šířen pod licencí IDPL , viz.[http://firebirdsql.org/]

[3] PostgreSQL je distribuován pod licencí BSD, jedná se o opensource projekt vývojové skupiny PostgreSQL Global Development Group, viz.[http://www.postgresql.org/]

Posledním, avšak neméně důležitým požadavkem, či spíše připomínkou, je zmínka o budoucím způsobu administrace a práce s databází.

Fakt, že bude využíváno rozhraní vytvořené pomocí PHP[4], může v mnohém značně ovlivnit podobu a funkčnost samotné databáze. Vzhledem k možnostem PHP při práci s databázemi je možné u některých činností zvolit zda se mají spustit na úrovni databáze nebo pomocí PHP, a u některých činností je dokonce výhodnější použít patřičné PHP skripty místo složitých funkcí, které nemusejí mít úplnou podporu v dané databázové platformě.

2.2. Varianty řešení

Na základě zadání a konzultací s vedoucím projektu jsem navrhl několik možných řešení, která více, či méně odpovídají popisované realitě. Jelikož cílem mé práce není porovnání jednotlivých variant řešení, ale samotný návrh databáze uvedu zde pouze dvě z možných variant návrhu. Popíši jejich výhody a nevýhody, objasním svojí motivaci pro vytvoření právě těchto dvou modelů a provedu vzájemné porovnání jednotlivých variant. Další řešení povětšinou vzniknou osamostatněním některých údajů do zvláštní tabulky, nebo naopak spojením několika tabulek do jedné.

[4] PHP-Hypertext Preprocessor je OpenSource projekt pod záštitou The PHP Group. Viz. [http://www.php.net/]

2.3. Varianta 1.

Tento model (viz. příloha schéma 1) byl navržen především s ohledem na co nejmenší redundanci dat, co největší přehlednost a jasné oddělení infrastrukturních prvků od prvků ostatních. Na tomto modelu je bohužel vidět negativní dopad snahy přizpůsobit databázi evidence infrastruktury pro účetní potřeby. Tento dopad se projevil rozdělením jedné tabulky na samostatnou tabulku zařízení a okolní tabulky s vazbou na tabulku zařízení, které obsahují podrobnější informace v závislosti na typu zařízení. Myšlenka samotného rozdělení je jistě v pořádku, jak se ukáže v dalším modelu, avšak toto konkrétní provedení nebylo nejšťastnějším.

Vzhledem k tomu, že majetek je v současné době evidován primárně ve vztahu k jeho umístění v místnostech a zároveň je pro ekonomické a účetní potřeby seskupen do skupin nejen podle typu, ale často i podle pořizovací ceny, je obtížné se v případě takovéhoto modelu vyrovnat s některými případy, které nastávají. Je snadné podle současné evidence majetku odhadnou, které dotazy na databázi budou zastoupeny nejčastěji. S největší pravděpodobností se bude jednat o reporty zařízení v daném umístění. O report zařízení s daným typem nebo cenou, o reporty zařízení pořízených za určité období, nebo pravděpodobně i o reporty softwaru ve vztahu k zařízením, na kterém jsou provozována a ve vztahu k pořizovací ceně.

Jedním z případů problémové evidence může být například evidence Harddisků v počítačích. Každý harddisk má logicky vazbu na určitý HW[5], avšak pokud je harddisk součástí HW v době pořízení, je součástí celku a tudíž se eviduje jako součást pořizovaného HW, avšak v případě, že se harddisk dokupoval později, má sice svoji vazbu na HW k němuž patří, ale z hlediska evidence majetku se jedná o samostatný prvek databáze na stejné úrovni jako samotný HW.

Jak je zřejmé, vyvstává zde problém, jak zajistit správné vazby mezi souvisejícími zařízeními a přitom ponechat databázi přehlednou, snadno udržovatelnou a hlavně lehce použitelnou.

[5]HW = "hardware." *The American Heritage® Science Dictionary*. Houghton Mifflin Company. 29 Dec. 2007. <Dictionary.com http://dictionary.reference.com/browse/hardware>.

„A computer, its components, and its related equipment. Hardware includes disk drives, integrated circuits, display screens, cables, modems, speakers, and printers. Compare software."

Uvažujme případ, že by nás zajímal soupis veškerého zařízení v určité místnosti. Potom musíme vybrat všechna zařízení s daným umístěním z tabulky zařízení. Pokud se ale chceme vyhnout případům, kdy se nám mezi PC bude potulovat několik harddisků, síťových karet, periferií a jiného zařízení evidovaného zvlášť, musíme přiřadit samostatně evidované objekty k patřičnému PC. Čím se nám samozřejmě dotaz na databázi stane složitějším a náročnějším.

Pokud by nás ovšem dále zajímaly podrobné informace o PC, museli bychom opět patřičně upravit náš dotaz. Tím se ovšem dotaz již stává poměrně komplikovaným a tudíž roste pravděpodobnost, že se objeví chyba v dotazu a následně i ve výsledku. Navíc takovýto dotaz zvyšuje nároky na databázi a HW na němž je databáze provozována, nemluvě o nárocích na tvůrce PHP rozhraní, který se s takovýmto dotazem musí následně vypořádat při jeho dalším zpracovávání.

Vzhledem ke složitosti evidence zařízení na Vyšší odborné škole je tento model použitelný, avšak s ohledem na komfort práce, vyhledávání, udržování a reportování jsem jej shledal nevhodným pro účely evidence infrastruktury na Vyšší odborné škole informačních služeb v Praze.

2.4. Varianta 2

Tato varianta (viz. příloha schéma 2)do jisté míry vychází z první varianty uvedené výše, avšak odstraňuje některé sporné a problémové vazby a některé tabulky integruje dohromady pro větší přehlednost a jednoduchost.

Hlavním cílem tohoto návrhu bylo vytvořit přehlednou a jednoduše použitelnou databázi pro evidenci infrastruktury a majetku. Důraz byl kladen hlavně na co nejmenší počet problémových vazeb, co největší přehlednost, použitelnost, snadnou rozšiřitelnost a jednoduchou administraci.

Vzhledem k faktu, že s databází budou jistě pracovat i běžní uživatelé, snažil jsem se návrh přizpůsobit především s ohledem na ně, jejich potřeby a komfort při jejich práci. Snažil jsem se dosáhnout cíle, kdy je možné většinu informací získat bez nutnosti použití složitých dotazů.

Snaha omezit nutnost použití složitých dotazů má kladný dopad i na tvorbu PHP rozhraní, kdy je snazší vytvoření insertů, selectů i updatů pro práci s databází prostřednictvím vytvořeného rozhraní. Díky tomu databáze i samotné PHP generují menší zátěž na HW s databází a snižuje se riziko případné chyby v dotazu nebo výsledku.

Na tomto modelu je již negativní dopad způsobený přizpůsobením databáze pro účetní evidenci minimalizován. Bylo toho dosaženo odlišným přístupem k řešení problému a rozdělením zařízení primárně podle jejich typu. Po přezkoumání požadavků jsem se rozhodl rozdělit evidovaný majetek do tří skupin a to na vybavení, infrastrukturní hardwarové zařízení a ostatní hardwarové zařízení. Toto rozdělení přináší mnohem větší přehlednost, snazší orientaci v databázi, menší počet vazeb a tabulek a díky tomu i snazší dotazy a administraci.

Všechna zařízení mohou mít mezi sebou vzájemnou vazbu přímo přes vazební tabulku, čímž se odstranil kupříkladu již zmiňovaný problém s evidencí harddisků. Díky jedné vazební tabulce a vazbám tabulek právě na ní, je možnost vzájemně propojit v podstatě jakékoli dvě tabulky nebo použít zpětnou vazbu na jednu jedinou tabulku.

3. Řešení

V této kapitole se zaměřím na konkrétní řešení, popis jednotlivých tabulek, objasnění proč je která hodnota právě toho nebo onoho datového typu, vysvětlení funkci číselníku a vazební tabulky, u některých hodnot a tabulek se zmíním o jejich alternativách, objasním své pohnutky právě k tomuto řešení a uvedu některé z dalších možností jako je použití funkcí a triggerů[6].

Jako řešení jsem vybral již zmiňovanou variantu 2, která i přes svou jednoduchost nejlépe reflektuje na zadané požadavky.

Vzhledem k požadavku použít volně dostupnou a použitelnou databázovou platformu jsem se rozhodl pro použití zřejmě nejrozšířenější databázové platformy, kterou je MySQL[7] od společnosti MySQL AB, která je k nekomerčním účelům poskytována zdarma pod licencí GPL[8].

Jako první krok je nutné vytvořit schéma[9], do kterého se budou naše tabulky tvořit a v němž budou uloženy. V našem případě jí pojmenujme například cmdb[10], toto pojmenování vychází z metodiky ITIL[11].

Jako znakovou sadu jsem zvolil dnes hojně používané UTF8[12].

---

[6] Trigger definuje činnost, která se provede v případě vzniku nějaké události.

[7] Viz. MySQL, http://cs.wikipedia.org/w/index.php?title=MySQL&oldid=1956380

[8] Viz. GNU General Public License, http://cs.wikipedia.org/w/index.php?title=GNU_General_Public_License&oldid=2090750 , nebo [http://www.gnu.org/]

[9] Databáze, nebo databázové schéma má v různých databázových platformách různá pojmenování. U produktů společnosti Microsoft jsou to databáze, společnost Oracle jim říká domény. Vždy jde ale o jednu stejnou věc. Je to prvek nadřazený jednotlivým tabulkám, které sdružuje do logického celku.

[10] CMDB = Configuration Management DataBase. Další informace o CMDB jsou dostupné například na adrese [http://en.wikipedia.org/wiki/CMDB]

[11] ITIL = Information Technology Infrastructure Library. Další informace dostupné na oficiálních stránkách [http://www.itil-officialsite.com/home/home.asp], nebo například na adrese anglické Wikipedie [http://en.wikipedia.org/wiki/Information_Technology_Infrastructure_Library]

[12] Znaková sada je součástí ISO normy ISO 10646-1:2000, další informace například na adrese [http://cs.wikipedia.org/wiki/UTF-8]

Samotné schéma vytvoříme pomocí následujícího SQL[13] příkazu

```
CREATE DATABASE `cmdb`
DEFAULT CHARACTER SET utf8 COLLATE utf8_czech_ci;
```

Kvůli předpokládané potřebě bezpečnosti dat bude výhodnější použít databázový engine InnoDB[14], který má plnou podporu transakcí[15]. InnoDB a podpora transakcí má bohužel negativní dopad na rychlost vyhledávání a práci s většími databázemi. Další možností by bylo použít MyISAM[16]. Databázový engine MyISAM je optimalizován přednostně pro prácí s velkým objemem dat. Na rozdíl od toho InnoDB je optimalizováno spíše pro bezpečnost.

Jazykovou sadu volit pro jednotlivé tabulky nemusíme, protože byla již vybrána při tvorbě samotné databáze. Tabulky tuto jazykovou sadu zdědí.

[13] SQL = Structured Query Language je dotazovaný jazyk používaný pro práci s databázemi. Viz. [http://cs.wikipedia.org/wiki/SQL], jazyk SQL je součástí ISO normy ISO/IEC 9075-13:2003

[14] Databázový engine od společnosti Innobase. Další informace viz. dokumentace MySQL [http://dev.mysql.com/doc/refman/5.0/en/innodb.html], nebo na domovských stránkách projektu InnoDB [http://www.innodb.com/]

[15] Transakce seskupuje sadu příkazů do jednoho celku a změny v databázi se projeví až po skončení průběhu celé transakce. V případě že během transakce dojde k chybě, provede se rollback. Transakce se zahájí příkazem Begin a potvrdí příkazem Commit, nebo zruším příkazem Rollback.

[16] Autorem databázového enginu MyISAM je společnost MySQL AB. Další informace na stránkách firmy MySQL AB na adrese [http://dev.mysql.com/doc/refman/5.0/en/myisam-storage-engine.html]

3.1. Návrh a popis jednotlivých tabulek databáze

V prvním kroku přistoupím k vytvoření tabulek, které jsou samostatné a nemají žádné vazby na okolní tabulky. Většinou se jedná o tabulky logů, tabulky číselníků, nebo například tabulky verzí a uživatelů. Některé samostatné tabulky uvedu až v průběhu tvorby jednotlivých tabulek tak aby logicky navazovaly na tabulky, se kterými budou logicky souviset. U tabulek které obsahují cizí klíč, uvedu nejprve hlavní tabulku a poté tabulku s cizím klíčem.

U každé tabulky ukážu, jak by mohly vypadat případné inserty do tabulky. V případě, že se bude jednat o složitější insert s vazbami na okolní tabulky, vysvětlím princip vkládání a vložení záznamu demonstruji postupem.

3.1.1. Tabulka log

Tabulka log slouží k logování změn v databázi.

Tabulka log je velice užitečným nástrojem monitoringu, jelikož často obsahuje přehled všech změn za určité časové období, může do jisté míry sloužit pro obnovu změněných nebo smazaných dat. Po drobné úpravě může sloužit i k logování uživatelů přihlášených do databáze.

K zaznamenávání informací do tabulky log lze použít PHP skript, nebo trigger. Vzhledem k faktu, že není definováno, jaké události by měli být logovány, je možná vhodnější provést logování pomocí PHP nad konkrétními tabulkami a hodnotami.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
nazev	VARCHAR (55)	Not null
akce	VARCHAR (20)	Not null
stara_hodnota	LONGTEXT	Not null
nova_hodnota	LONGTEXT	Not null
zmenil	VARCHAR(20)	Not null
zmeneno	DATE	Not null

- ID je primární klíč tabulky, jako takový nemůže mít nikdy hodnotu null, nenabývá záporných hodnot a má nastaven atribut auto increment, což znamená, že s každým dalším vloženým záznamem se ID automaticky zvyšuje o jedna.
- NAZEV obsahuje název záznamu, jehož se změna týkala.
- AKCE je reprezentována popisem akce, například insert, delete, ….
- STARA a NOVA HODNOTA obsahují hodnoty před a po změně záznamu.
- ZMENIL obsahuje login uživatele, který změnu provedl.
- ZMENENO je datum změny záznamu v databázi.

```
CREATE TABLE `cmdb`.`log`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `nazev` VARCHAR( 55 ) NOT NULL ,
 `akce` VARCHAR( 20 ) NOT NULL ,
 `stara_hodnota` LONGTEXT NOT NULL ,
 `nova_hodnota` LONGTEXT NOT NULL ,
 `zmenil` VARCHAR( 20 ) NOT NULL ,
 `zmeneno` DATE NOT NULL ,
 PRIMARY KEY (`id`)
)
TYPE = innodb;
```

Příklad insertu do tabulky logu

```
INSERT
INTO`log`(`id`,`nazev`,`akce`,`stara_hodnota`,`nova_hodnota`,`zmenil`,
`zmeneno`)
VALUES (NULL,'Server','Update','10.10.0.1','255.255.255.255','lsipka',
'2007-12-29'),
(NULL,'Aplikace','Delete','','','lsipka','2007-12-29')
```

Takto by mohl vypadat insert do tabulky `log`.

- Na první hodnotě vidíme, že došlo k updatu ip adresy u zařízení s názvem „Server“, tato změna byla provedena 29.12.2007 a provedl jí uživatel přihlášený jako „lsipka“.
- Druhý insert demonstruje podobu záznamu v případě, že došlo ke smazání nějakých dat. V našem případě došlo ke smazání záznamu s názvem „Aplikace“, smazána byla 29.12.2007 a delete provedl uživatel „lsipka“.

3.1.2. Tabulka ciselnik

Číselník je zvláštním typem tabulky. Jak již název napovídá, číselník obsahuje v jednom sloupci číslo, které slouží jako zástupný prvek pro data, která se často opakují, nebo nejsou vhodná pro vkládání do normálních tabulek. Často bývá použito číselníků více, jeden pro každý typ dat. V našem případě máme pouze jednu tabulku číselníků, která obsahuje všechny číselníkové hodnoty, které by jinak byly ve velkém množství samostatných tabulek. Toto řešení je vhodné hlavně v případech, kdy máme větší množství číselníků, ale menší počet jednotlivých číselníkových záznamů.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
nazev	VARCHAR (255)	Not null, Unique
typ	VARCHAR (55)	Not null, Unique
popis	VARCHAR (55)	Not null
int1	INTEGER	Null
var1	VARCHAR(255)	Null
text1	LONGTEXT	Null

- ID je v tabulce číselníku primárním klíčem.
- NAZEV obsahuje název záznamu číselníku
- TYP rozlišuje jednotlivé typy číselníků, jde o unikátní položku v tabulce.
- POPIS obsahuje textový popis číselníkové hodnoty, spolu se záznamem TYP tvoří složený unikátní index, jehož hodnoty se smějí v tabulce vyskytnout pouze jednou.
- INT1, VAR1, TEXT1 jsou nepovinné položky, které slouží jako doplňující pro další informace k číselníku.

```
CREATE TABLE `cmdb`.`ciselnik`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `nazev` VARCHAR( 255 ) NOT NULL ,
 `typ` VARCHAR( 55 ) NOT NULL ,
 `popis` VARCHAR( 55 ) NOT NULL ,
 `int1` INT NOT NULL ,
 `var1` VARCHAR( 255 ) NOT NULL ,
 `text1` LONGTEXT NOT NULL ,
 PRIMARY KEY (`id`),
 UNIQUE (`typ`,`popis`)
)
TYPE = innodb;
```

Příklad insertů do tabulky číselníků

Insert do tabulky `ciselnik` je trošku komplikovanější. Je to způsobeno použitím jednoho univerzálního číselníku, namísto většího množství menších a jednodušších tabulek číselníků.

Než začneme vkládat jednotlivé číselníky, je vhodné si rozmyslet, jaký rozsah bude každý číselník zabírat, tj. v jakém číselném rozmezí bude který číselník mít svoje data. V našem

příkladě budeme uvažovat rozsah násobků 100. Praxi to znamená, že první číselník bude mít vymezen prostor mezi 1-99, druhý 100-199 atd.

Náš první číselník bude vlastně číselník číselníků. Tj. rozsah 1-99 bude obsazen seznamem použitých číselníků. Toho docílíme tím, že první hodnota, kterou vložíme, bude mít jiný tvar sql insertu.

Na pozici NAZEV dáme budoucí typ číselníku, na pozici TYP bude textová hodnota určená speciálně pro tento seznam číselníků a na pozici POPIS vložíme popis číselníku.

```
INSERT INTO `ciselnik`
(`id`,`nazev`,`typ`,`popis`,`int1`,`var1`,`text1`)
VALUES (1,'priorita','special','Číselník priorit',0,'','').
```

Tímto jsme vytvořily první záznam v číselníku, který má TYP = special a NAZEV = priorita.

Jako další krok vložíme do našeho číselníku tři hodnoty pro malou, střední a vysokou prioritu. V tuto chvíli je vhodné ohlídat si hodnotu ID. Do rozsahu 0-99 nic vložit nemůžeme, ten již byl obsazen pro číselníky typu special, proto zvolíme další rozsah k dispozici, a tím je 100-199.

```
INSERT INTO `ciselnik`
(`id`,`nazev`,`typ`,`popis`,`int1`,`var1`,`text1`)
VALUES
(100,'Vysoká','priorita','Vysoká priorita',1,'',''),
(101,'Střední','priorita','Střední priorita',2,'',''),
(102,'Nízká','priorita','Nízká priorita',3,'','');
```

Zvolený číselný rozsah nebyl čistě náhodný. Pokud se pozorně zadíváme na ID hodnot v databázi, snadno poznáme, že záznam s typem special nám svým ID ukazuje na číselný rozsah hodnot daného typu. Tudíž náš speciální záznam na ID = 1 svým ID určuje, že jeho číselná řada je na pozicích 100 - 199.

Obdobě bude speciální záznam s ID = 2 odkazovat svým ID na číselný rozsah jeho hodnot, tj. 200 – 299, nebo záznam s ID = 25 bude odkazovat na řadu 2500-2599 atd.

```
INSERT INTO `ciselnik`
(`id`,`nazev`,`typ`,`popis`,`int1`,`var1`,`text1`)
VALUES (2,'akce','special','Číselník akcí',0,'','');
```

```
INSERT INTO `ciselnik`
(`id`,`nazev`,`typ`,`popis`,`int1`,`var1`,`text1`)
VALUES
```

```
(200,'Delete','akce','Akce delete',0,'',''),
(201,'Update','akce','Akce update',0,'',''),
(202,'Insert','akce','Akce insert',0,'',''),
(203,'Select','akce','Akce select',0,'','');
```

Pro evidenci seznamu vazeb a pro rychlejší vyhledávání je výhodné, buďto založit tabulku databáze podobnou níže uvedené, nebo si vést podobnou evidenci např. pomocí nějaké tabulky v příslušném programu. Vedení podobného souboru ať již formou tabulky v databázi, nebo jiným způsobem má hlavní výhodu v přehledném seznamu rozsahů ID, jejich typů a pak hlavně hodnot ve sloupcích INT1, VAR1 a TEXT1, kde může mít každý číselník jiné hodnoty.

Například u typu priorita jsme do sloupce INT1 vložili číslo, které dále určuje prioritu. Tudíž vysoká priorita je priorita 1 s ID = 100, střední priorita je priorita 2 s ID = 101, atd.

Rozsah ID	Typ	Popis	INT1	VAR1	TEXT1
1 - 99	special	Číselník záznamů typu special	Hodnota	Hodnota	Hodnota
100 - 199	priorita	Číselník priorit	Hodnota	Hodnota	Hodnota
200 - 299	akce	Číselník akcí	Hodnota	Hodnota	Hodnota

Položky číselníků z tabulek databáze

Tabulka	Položka
Kontakty	skupina
Žádosti	typ
Žádosti	akce
Žádosti	priorita
Aplikace	typ
Vybavení	typ
Other_HW	typ
HW	typ

3.1.3. Tabulka kontakty

Tabulka `kontakt`, jak již její název značí, bude obsahovat informace o kontaktech. V našem případě to budou informace o zaměstnancích. Na tabulku kontakty navazuje tabulka `users`.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
jmeno	VARCHAR(55)	Not null, Unique
prijmeni	VARCHAR(55)	Not null, Unique
titul	VARCHAR(15)	Null
email	VARCHAR(55)	Not null, Unique
telefon	INTEGER	Not null
skupina	INTEGER	Null
poznamka	LONGTEXT	Null

- ID je primární identifikátor tabulky kontakty.
- JMENO, PRIJMENI, TITUL, EMAIL, TELEFON jsou hodnoty, které obsahují informace ke kontaktům. Hodnoty JMENO, PRIJMENI, EMAIL tvoří složený unikátní klíč/index.
- SKUPINA je hodnota z číselníku určující, které skupině osoba náleží.
- POZNAMKA může obsahovat další dodatečné informace ke kontaktu.

```
CREATE TABLE `kontakty`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `jmeno` VARCHAR( 55 ) NOT NULL ,
 `prijmeni` VARCHAR( 55 ) NOT NULL ,
 `titul` VARCHAR( 15 ) ,
 `email` VARCHAR( 55 ) NOT NULL ,
 `telefon` INT NOT NULL ,
 `skupina` INT,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` ) ,
 UNIQUE (`jmeno`,`prijmeni`,`email`)
)
TYPE = innodb;
```

Příklad insertu do tabulky kontaktů[17]

Insert kontaktů by neměl skýtat žádná úskalí.

```
INSERT INTO `kontakty` ( `id` , `jmeno` , `prijmeni` , `titul` ,
`email` , `telefon` , `skupina` , `poznamka` )
VALUES
(NULL, 'Testovací', 'Subjekt 1', NULL , 'test1@domain.com',
'123456789', 0, 'Poznámka k prvnímu testovacímu kontaktu'),
(NULL, 'Testovací', 'Subjekt 2', 'Ing', 'test2@domain.com',
'987654321', 0, 'Poznámka k druhýmu testovacímu kontaktu')
```

[17] Jelikož se jedná o demonstraci, kdy nemáme k dispozici skutečná data pro číselníky, bude v insertu pod položkou SKUPINA číslo 0. Se skutečnými daty by tam patřila patřičná položka z tabulky číselníků.

3.1.4. Tabulka users

Tabulka vznikla čistě za účelem potřeby evidence přístupů, přihlášení a oprávnění osob, které přistupují k databázi přes rozhraní PHP. Samotné PHP Neumí rozhodovat o právech osob přihlášených pro práci s databází, proto nám tato tabulka dává možnost nastavit v závislosti na právech osob v tabulce `users` práva pro určité akce na úrovni PHP pomocí jednoduchých podmínek. Tato práva jsou nastavena v položkách AKCE. Tabulka `users` má vazbu na tabulku kontaktů a obsahuje jeden cizí klíč právě z tabulky kontaktů.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
id_kont	INTEGER	Cizí klíč z tabulky kontakty, Not null, Unique
login	VARCHAR(55)	Not null, Unique
pass	VARCHAR(15)	Not null
akce1	TINYINT(1)	Not null
akce2	TINYINT(1)	Not null
poznamka	LONGTEXT	Null

- ID je unikátní identifikátor řádku tabulky.
- ID_KONT je cizí klíč z tabulky kontakty. Zde může i nemusí být povolena null hodnota, stejně jako může a nemusí být vazba mezi tabulkou `kontakty` a tabulkou `users`. Pokud předpokládáme, že k naší databázi budou přistupovat pouze lidé zevnitř organizace, je pravděpodobné že budou mít záznam v tabulce kontakty. V takovémto případě není důvod ponechávat hodnotu ID_KONT prázdnou a rušit vazbu již zmíněných dvou tabulek `kontakty` a `users`. Pokud ovšem bude s databází manipulovat i osoba mimo organizaci, záleží vše na tom zda, bude osoba v tabulce `kontakty`.
- LOGIN obsahuje login uživatele pro přihlášení pomocí PHP.
- PASS je heslo uživatele přistupujícího přes rozhraní PHP. Pomocí funkcí PHP lze heslo ukládat do databáze jako hash18 a provádět kontrolu pomocí porovnání, nebo ho lze ukládat v běžné čitelné podobě.
- AKCE představuje boolean datový typ, tedy hodnoty 1 x 0, či např. true x false, ano x ne. V závislosti na tom zda bude hodnota 1, nebo 0 bude, nebo nebude mít uživatel oprávnění k provedení příslušné akce.
- POZNAMKA slouží pro další doplňující informace.

[18] Viz. též encyklopedie Wikipedie, heslo Hashovací funkce, dostupné na adrese [http://cs.wikipedia.org/wiki/Hashovac%C3%AD_funkce].

```
CREATE TABLE `users`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `id_kont` INT UNSIGNED NOT NULL ,
 `login` VARCHAR( 55 ) NOT NULL ,
 `pass` VARCHAR( 55 ) NOT NULL ,
 `akce1` TINYINT( 1 ) NOT NULL ,
 `akce2` TINYINT( 1 ) NOT NULL ,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` ) ,
 UNIQUE (`id_kont`,`login`),
CONSTRAINT `kontakt_user` FOREIGN KEY `kontakt_user` ( `id_kont` )
REFERENCES `kontakty` ( `id` ) ON DELETE RESTRICT ON UPDATE RESTRICT
)
TYPE = innodb;
```

Příklad insertu do tabulky uživatelů

V podstatě jediná věc, na kterou je nutné si dát při ukládání dat do tabulky `users` pozor je fakt, že obsahuje cizí klíč z tabulky kontaktů.

```
INSERT INTO `users` ( `id` , `id_kont` , `login` , `pass` , `akce1` ,
`akce2` , `poznamka` )
VALUES
(NULL, '1', 'test1', 'test1', '0', '0', 'Poznámka 1'),
(NULL, '2', 'test2', MD5( 'test2' ) , '1', '1', 'Poznámka')
```

Jak je vidět vložili jsme do tabulky dva záznamy. Oba pomocí ID_KONT ukazují do tabulky kontaktů na příslušný kontakt. Dále je vidět, že první uživatel má pro akce 1 a 2 nastavenou hodnotu 0, z čehož lze usuzovat, že nebude mít oprávnění tyto akce provádět. Nakonec je vhodné si všimnout, že první uživatel má heslo vložené jako obyčejný text, zatímco druhý uživatel má heslo kryptované.

3.1.5. Tabulka denik

Tabulka deník slouží k zaznamenání událostí minulých, ale i budoucích. Ve vazbě na kontakty jej lze použít například pro plánování dovolené, při vazbě na lokality například k informování o plánovaných opravách a nedostupnosti učeben.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
typ_udalosti	VARCHAR (55)	Not null
udalost	LONGTEXT	Not null
datum_od	DATETIME	Null
datum_do	DATETIME	Null
poznamka	LONGTEXT	Null

- ID je primárním klíčem a identifikátorem položek v tabulce `denik`
- TYP_UDALOSTI je číselníková položka která charakterizuje, jaký druh činnost se jedná.
- UDALOST obsahuje popis činnosti.
- DATUM_OD a DATUM_DO ohraničují časově dobu konání činnosti.
- POZNAMKA je pro další doplňující informace k události.

```
CREATE TABLE `denik`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `typ_udalosti` VARCHAR (55) NOT NULL ,
 `udalost` LONGTEXT NOT NULL ,
 `datum_od` DATETIME,
 `datum_do` DATETIME,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` )
)
TYPE = innodb;
```

U položky TYP_UDALOSTI je možné místo vkládání textu použít položky číselníků.

Příklad insertu do tabulky deník

Vzhledem k jednoduchosti tabulky se jedná o zcela běžný insert.

```
INSERT INTO `denik` ( `id` , `typ_udalosti` , `udalost` , `datum_od` ,
`datum_do` , `poznamka` )
VALUES
(NULL, 'Dovolená', 'Tento kontakt má zřejmě dovolenou', '2007-12-23
02:05:17', '2007-12-30 01:05:22', NULL),
(NULL, 'Migrace', 'Jedná se o plánovanou migraci aplikací na
zařízení', '2007-12-30 02:05:53', '2007-12-30 02:35:57', NULL)
```

Jak je patrné z insertu vložili jsme do deníku dvě události. První událostí je dovolená nějakého zaměstnance, druhá nás informuje o tom, že proběhne migrace nějaké aplikace z vývojového/testovacího prostředí na produkční, tudíž je pravděpodobné, že bude aplikace nedostupná.

3.1.6. Tabulka vazba

Jde o vazební tabulku, která navzájem propojuje ostatní tabulky. Princip vazební tabulky je ve vzájemném propojení jednotlivých tabulek pro snazší práci s nimi.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
f_key1	INTEGER	Not null, Unique
f_key2	INTEGER	Not null, Unique
typ_vazby	INTEGER	Not null, Unique
popis_vazby	VARCHAR(255)	Not null
poznamka	LONGTEXT	Null
int1	INTEGER	Null
var1	VARCHAR(255)	Null
text1	LONGTEXT	Null

- ID je primární klíč
- F_KEY1 a F_KEY2 jsou ID z okolních tabulek, přes které se provádí vazba.
- TYP_VAZBY určuje, o jaký typ vazby se jedná. Spolu s oběma hodnotami F_KEY tvoří unikátní složený klíč/index, který se může vyskytovat v celé tabulce pouze jednou.
- POPIS_VAZBY obsahuje slovní popis vazby, nebo typu vazby.
- POZNAMKA může obsahovat další doplňující informace k vazbě
- INT1, VAR1, TEXT1 jsou použitelné pro další doplňující informace k vazbě.

```
CREATE TABLE `cmdb`.`vazba`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `f_key1` INT UNSIGNED NOT NULL ,
 `f_key2` INT UNSIGNED NOT NULL ,
 `typ_vazby` INT UNSIGNED NOT NULL ,
 `popis_vazby` VARCHAR( 255 ) NOT NULL ,
 `poznamka` LONGTEXT,
 `int1` INT,
 `var1` VARCHAR( 255 ) ,
 `text1` LONGTEXT,
 PRIMARY KEY ( `id` ) ,
 UNIQUE (`f_key1`,`f_key2`,`typ_vazby`)
)
TYPE = innodb;
```

Jak již bylo řečeno výše, tabulka `vazba` obsahuje seznam všech vzájemných vazeb tabulek. Založení vazby, ani její pochopení a interpretace neskýtá žádná úskalí, jako například předchozí tabulka číselníků.

Příklad insertu a selectu do/z vazební tabulky

Pro demonstraci vazby vazební tabulky provedeme vazbu mezi tabulkou s kontakty a tabulkou deníku.

```
INSERT INTO `vazba` ( `id` , `f_key1` , `f_key2` , `typ_vazby` ,
`popis_vazby` , `poznamka` , `int1` , `var1` , `text1` )
VALUES
(NULL, '2', '1', '1', 'kontakt-denik', 'Toto je vazba mezi tabulkou
Kontakty a tabulkou Denik', NULL , NULL , NULL)
```

Z insertu je vidět, že F_KEY1 je ID z tabulky `kontakty` a F_KEY2 je ID z tabulky `denik`. Vazba byla založena jako první, proto má TYP_VAZBY = 1. Hodnota typu vazby ovšem může být libovolná, pro přehlednost ale doporučuji používat stoupající číselnou řadu.

Chceme-li nyní zjistit kdy přesně má určitý kontakt dovolenou, můžeme položit například takovýto dotaz.

```
select k.jmeno, k.prijmeni, d.typ_udalosti,d.datum_od,d.datum_do
from vazba v join kontakty k
on k.id=v.f_key1
left join denik d on d.id=v.f_key2
where k.id=2
```

Obdobným způsobem zle vytvořit vazby mezi libovolnými dvěma tabulkami v databázi, které jsou napojené na vazební tabulku. Lze tak získat například vazby mezi HW a aplikacemi, mezi HW a kontakty, mezi lokalitou a HW a další.

3.1.7. Tabulka typ_vazby

Tabulka typ vazby obsahuje přehled a popis jednotlivých vazeb vazební tabulky `vazba`. Tato tabulka je obdobou doporučované tabulky přehledů pro číselníky. Pomocí této tabulky lze dosáhnout mnohem většího přehledu o tom, které vazby obsahuje vazební tabulka a jaký význam mají jaké hodnoty.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
f_key1	VARCHAR(55)	Not null
f_key2	VARCHAR(55)	Not null
typ_vazby	INTEGER	Not null
int1	VARCHAR(55)	Null
var1	VARCHAR(55)	Null
text1	VARCHAR(55)	Null

- ID je primární klíč tabulky
- F_KEY1 určuje z jaké tabulky je první cizí klíč v tabulce `vazba`.
- F_KEY2 určuje z jaké tabulky je druhý cizí klíc v tabulce `vazba`.
- TYP_VAZBY obsahuje číselnou hodnotu typu vazby.
- INT1 je popis hodnoty v tabulce `vazba` ve sloupci INT1 právě u této konkrétní vazby.
- VAR1 je popis hodnoty v tabulce `vazba` ve sloupci VAR1 této konkrétní vazby.
- TEXT1 popisuje význam hodnoty v tabulce `vazba` ve sloupci TEXT1 této vazby.

```
CREATE TABLE `typ_vazby`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `f_key1` VARCHAR( 55 ) NOT NULL ,
 `f_key2` VARCHAR( 55 ) NOT NULL ,
 `typ_vazby` INT UNSIGNED NOT NULL ,
 `int1` VARCHAR( 55 ) ,
 `var1` VARCHAR( 55 ) ,
 `text1` VARCHAR( 55 ) ,
 PRIMARY KEY ( `id` )
)
TYPE = innodb;
```

Příklad insertu do tabulky typů vazby

Použijeme již vloženou vazbu mezi kontaktem a deníkem. Jedná se o jednoduchý insert, který neobsahuje žádné záludnosti.

```
INSERT INTO `typ_vazby` ( `id` , `f_key1` , `f_key2` , `typ_vazby` ,
`int1` , `var1` , `text1` )
VALUES
(NULL, 'kontakty', 'denik', '1', 'Vyznam hodnoty INT1', 'Vyznam
hodnoty VAR1', 'Vyznam hodnoty TEXT1')
```

3.1.8. Tabulka lokalita

Tabulka lokalit obsahuje seznam a popis lokalit, místností a umístění jednotlivých infrastrukturních prvků a majetku. V našem případě se jedná o tabulku velice podobnou klasickému číselníku, tudíž by bylo možné tuto tabulku spojit s tabulkou `ciselnik`, avšak z důvodů velkého množství informací, které je možné k lokalitám evidovat, je vhodnější použít samostatnou tabulku.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
mistnost	VARCHAR(20)	Not null
patro	INTEGER	Null
poznamka	LONGTEXT	Null

- ID je primární klíč tabulky.
- MISTNOST i PATRO odkazují na konkrétní umístění zařízení
- POZNAMKA může obsahovat další doplňující informace.

```
CREATE TABLE `lokalita`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `mistnost` VARCHAR(20) NOT NULL ,
 `patro` INT,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` )
)
TYPE = innodb;
```

Příklad insertu do tabulky lokalit

Do naší tabulky lokalit vložíme dvě lokality, nebo přesněji místnosti.

```
INSERT INTO `lokalita` ( `id` , `mistnost` , `patro` , `poznamka` )
VALUES
(NULL, '1.7', '1', 'Servrovna'),
(NULL, '1.7', '1', 'p. Skopec')
```

Tabulka lokalit nyní obsahuje dvě umístění a to místnost 1.7, která je ale rozdělena na dvě části. Jejich rozlišení lze provést pomocí POZNAMKY nebo přímo pomocí MISTNOST. Já jsem se rozhodl pro první variantu.

3.1.9. Tabulka rack

Tabulka rack je velice podobná tabulce lokalita, obsahuje totiž také umístění zařízení, avšak v tomto případě již přesné umístění v podobě konkrétního umístění v racku včetně jeho pozice, která může být uložena ve vazební tabulce například jako hodnota INT1.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
nazev	VARCHAR(20)	Null
ic	VARCHAR(20)	Not null, Unique
lokalita	INTEGER	Not null
vyrobce	VARCHAR(20)	Null
model	INTEGER	Null
počet_pozic	INTEGER	Null
datum_porizeni	DATE	Not null
cena	FLOAT	Not null
poznamka	LONGTEXT	Null

- ID je primární klíč tabulky
- NAZEV je nevinná položka, která obsahuje pojmenování racku kvůli snazší evidenci
- IC je inventární číslo racku, je unikátní nejen v rámci tabulky racků ale napříč celou databází
- LOKALITA je v podstatě cizím klíčem z tabulky Lokalita.
- VYROBCE, MODEL, POCET_POZIC obsahují dodatečné informace ke každému racku.
- DATUM_PORIZENI představuje datum, kdy byl rack zakoupen, objednán, nebo zaplacen.
- CENA, tento údaj je důležitý pro účetnictví a představuje cenu zakoupeného zařízení.
- POZNAMKA obsahuje již tradičně další doplňující informace k racku.

```
CREATE TABLE `rack`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `nazev` VARCHAR( 20 ) ,
 `ic` VARCHAR( 20 ) NOT NULL ,
 `lokalita` INT NOT NULL ,
 `vyrobce` VARCHAR( 20 ) ,
 `model` INT,
 `pocet_pozic` INT UNSIGNED,
 `datum_porizeni` DATE NOT NULL ,
 `cena` FLOAT NOT NULL ,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` ) ,
 UNIQUE (`ic`)
)
TYPE = innodb;
```

Příklad insertu do tabulky racků

Do tabulky racků vložíme jeden rack, který bude umístěn v servrovně v místnosti 1.7. Jak je zřejmé, hodnota pro položku LOKALITA je ID z tabulky `lokality`.

```
INSERT INTO `rack` ( `id` , `nazev` , `ic` , `lokalita` , `vyrobce` ,
`model` , `pocet_pozic` , `datum_porizeni` , `cena` , `poznamka` )
VALUES
(NULL, 'Rack 1', 'ABC123', 1, NULL , NULL , 42 , '2007-12-01',
'120000', NULL)
```

3.1.10. Tabulka zadosti

Tabulka zadosti není zcela typickou tabulkou pro infrastrukturní databázi i když částečně s infrastrukturou souvisí. Jedná se o tabulku pro evidenci žádostí uživatelů databáze na administrátora databáze či infrastruktury. Velká výhoda této tabulky spočívá v možnosti přehledného třídění žádostí podle priority, druhu požadované akce, nebo pro zobrazení již zpracovaných žádostí a statické reportování doby potřebné k řešení a další informace.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
typ	INTEGER	Not null
akce	INTEGER	Not null
pozadano	DATE	Not null
vyrizeno	DATE	Null
priorita	INTEGER	Null
poznamka	LONGTEXT	Null

- ID je primární klíč tabulky.
- TYP je číselníková položka obsahující informace o typu žádosti.
- AKCE je také číselníková položka, která reprezentuje akci která je předmětem žádosti.
- POZADANO a VYRIZENO obsahují datum, kdy byla podána, nebo vyřízena žádost.
- PRIORITA je hodnota z číselníku, značící o jak urgentní žádost se jedná.
- POZNAMKA slouží uložení dalších doplňujících informací.

```
CREATE TABLE `zadosti`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `typ` INT NOT NULL ,
 `akce` INT NOT NULL ,
 `pozadano` DATE NOT NULL ,
 `vyrizeno` DATE,
 `priorita` INT,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` )
)
TYPE = innodb;
```

Příklad insertu do tabulky žádostí[19]

Do tabulky pro představu vložíme jednu žádost s vysokou prioritou. Pomocí vazební tabulky bychom pak mohli žádost dále navázat na patřičný HW a kontakt. Na místo hodnoty PRIORITA můžeme vložit buď ID vysoké priority z číselníků, nebo hodnotu 1, které je v popisu číselníkové položky „vysoká priorita“. Jen musíme být opatrní u případných selectů nezapomenou patřičně upravit podmínku.

```
INSERT INTO `zadosti` ( `id` , `typ` , `akce` , `pozadano` ,
`vyrizeno` , `priorita` , `poznamka` )
VALUES (NULL, 0, '200', '2007-12-22', NULL , '100', NULL)
```

[19] Jelikož se jedná o demonstraci, kdy nemáme k dispozici skutečná data pro číselníky, bude v insertu pod položkou TYP číslo 0. Se skutečnými daty by tam patřila patřičná položka z tabulky číselníků.

3.1.11. Tabulka aplikace

Tabulka aplikace slouží k udržování přehledu o zakoupených a provozovaných aplikacích, o jejich ceně, o platnosti jejich licence i o počtu zakoupených licencí.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
nazev	VARCHAR(55)	Not null
typ	INTEGER	Not null
verze	VARCHAR(20)	Null
ic	VARCHAR(20)	Not null, Unique
pocet	INTEGER	Null
popis	LONGTEXT	Null
cena	FLOAT	Not null
platnost_do	DATE	Null
porizeno	DATE	Not null
poznamka	LONGTEXT	Null

- ID je primární klíč tabulky.
- NAZEV obsahuje název aplikace.
- TYP je číselníková hodnota, která určuje, o jaký typ aplikace se jedná.
- VERZE obsahuje verzi aplikace nebo softwaru.
- IC je inventární číslo, které je unikátní v rámci celé databáze.
- POCET je počet licencí daného softwaru či aplikace.
- POPIS je textový popis aplikace, který doplňuje další informace k aplikaci.
- CENA je pořizovací cena aplikace nebo softwaru.
- PLATNOST_DO je platnost licence softwaru nebo aplikace.
- PORIZENO obsahuje datum, kdy byl software nebo aplikace pořízena.
- POZNAMKA může obsahovat další doplňující informace.

```
CREATE TABLE `aplikace`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `nazev` VARCHAR( 55 ) NOT NULL ,
 `typ` INT UNSIGNED NOT NULL ,
 `verze` VARCHAR( 20 ) ,
 `ic` VARCHAR( 20 ) NOT NULL ,
 `pocet` INT,
 `popis` LONGTEXT,
 `cena` FLOAT NOT NULL ,
 `platnost_do` DATE,
 `porizeno` DATE NOT NULL ,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` ) ,
 UNIQUE (`ic`)
)
TYPE = innodb;
```

Příklad insertu do tabulky aplikací[20]

Do tabulky vložíme aplikaci GIMP, která je k dispozici pod licencí GPL a proto má nulovou cenu, nulový počet licencí i nulovou platnost licence. Jako druhý záznam vložíme aplikace Photoshop od společnosti Adobe® v počtu 10 licencí s platností do 1.1.2010.

```
INSERT INTO `aplikace` ( `id` , `nazev` , `typ` , `verze` , `ic` ,
`pocet` , `popis` , `cena` , `platnost_do` , `porizeno` , `poznamka` )
VALUES
(NULL, 'Gimp', '0', '2.3', 'ABC123', 0, 'Graficky software.', '0',
'0000-00-00', '2007-12-01', NULL),
(NULL, 'Adobe Photoshop', '0', '12', 'ADOBE123', 10, 'Graficky
software', '15000', '2010-01-01', '2001-01-01', NULL)
```

[20] Jelikož se jedná o demonstraci, kdy nemáme k dispozici skutečná data pro číselníky, bude v insertu pod položkou TYP číslo 0. Se skutečnými daty by tam patřila patřičná položka z tabulky číselníků.

3.1.12. Tabulka vybaveni

Tabulka vybavení slouží primárně k evidenci majetku. Její potřeba vychází z potřeby použití databáze nejen pro evidenci infrastrukturních prvků, ale i pro evidenci veškerého majetku, který je v současné době nějakým způsobem evidován a někde vykazován. V tabulce vybavení by měl být všechen majetek, který není hardwarem, nebo infrastrukturním prvkem.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
typ	INTEGER	Not null
druh	VARCHAR(255)	Null
nazev	VARCHAR(20)	Not null
sn	VARCHAR(55)	Null
ic	VARCHAR(20)	Not null, Unique
lokalita	INTEGER	Not null
prechodna_lokalita	INTEGER	Null
cena	FLOAT	Not null
datum_porizeni	DATE	Not null
poznamka	LONGTEXT	Null

- ID je primární klíč tabulky
- TYP je hodnota z číselníků, které blíže určuje, o jaké jde zařízení.
- DRUH je nepovinná textová položka, která může obsahovat bližší popis zařízení.
- NAZEV je název zařízení.
- SN je sériové číslo zařízení
- IC je inventární číslo zařízení, které je unikátní napříč celou databází.
- UMISTENI odkazuje na tabulku `lokalita` a na umístění zařízení.
- PRECHODNE_UMISTENI může také odkazovat na tabulku `lokalita` a používá se především, pokud je na přechodnou dobu zařízení umístěno na jiném místě, než na jakém je evidováno.
- CENA je pořizovací cena zařízení.
- DATUM_PORIZENI je datum kdy bylo zařízení zakoupeno.
- POZNAMKA může obsahovat doplňující informace k záznamu nebo zařízení.

```
CREATE TABLE `vybaveni`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `typ` INT NOT NULL ,
 `druh` VARCHAR( 255 ) NOT NULL ,
 `nazev` VARCHAR( 20 ) ,
 `sn` VARCHAR( 55 ) NOT NULL ,
 `ic` VARCHAR( 20 ) NOT NULL ,
 `lokalita` INT NOT NULL ,
 `prechodna_lokalita` INT,
 `cena` FLOAT NOT NULL ,
 `datum_porizeni` DATE NOT NULL ,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` ) ,
 UNIQUE (`ic`)
)
TYPE = innodb;
```

Příklad insertu do tabulky vybavení[21]

Jako ukázku insertů vložíme do databáze skříň, která je v místnosti 1.7 v servovně, ale přechodně je umístěna, kvůli dočasnému nedostatku prostoru, u p. Skopce.

```
INSERT INTO `vybaveni` ( `id` , `typ` , `druh` , `nazev` , `sn` , `ic`
, `lokalita` , `prechodna_lokalita` , `cena` , `datum_porizeni` ,
`poznamka` )
VALUES
(NULL, '0', 'skříň malá', 'Malá skříň', 'SN123', '123456789', '1',
'2', '1000', '2007-12-30', 'Hezká malá skříň')
```

[21] Jelikož se jedná o demonstraci, kdy nemáme k dispozici skutečná data pro číselníky, bude v insertu pod položkou TYP číslo 0. Se skutečnými daty by tam patřila patřičná položka z tabulky číselníků.

3.1.13. Tabulka other_hw

Tato tabulka obsahuje ostatní infrastrukturní zařízení a hardware, který není z nějakého důvodu možno vložit do tabulky hw. Do této tabulky bude například patřit náš, již zmiňovaný, později dokupovaný harddisk.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
typ	INTEGER	Not null
druh	VARCHAR(255)	Null
nazev	VARCHAR(20)	Not null
sn	VARCHAR(55)	Null
ic	VARCHAR(20)	Not null, Unique
lokalita	INTEGER	Not null
prechodna_lokalita	INTEGER	Null
velikost	FLOAT	Null
model	VARCHAR(20)	Null
vyrobce	VARCHAR(20)	Null
cena	FLOAT	Not null
datum_porizeni	DATE	Not null
popis	LONGTEXT	Null
poznamka	LONGTEXT	Null

- ID je primární identifikátor tabulky
- TYP je číselníková položka, která blíže určuje, o jaké zařízení se jedná.
- DRUH je textová položka, která slouží k podrobnějšímu popisu zařízení.
- NAZEV je název zařízení.
- SN je sériové číslo zařízení.
- IC je inventární číslo unikátní pro celou databázi.
- LOKALITA označuje umístění zařízení pomocí vazby na tabulku `lokalita`.
- PRECHODNA_LOKALITA značí dočasné umístění zařízení.
- VELIKOST slouží v případě zařízení, u nichž má tento atribut smysl, k záznamu velikost zařízení.
- MODEL a VYROBCE slouží k zaznamenání výrobce a modelu zařízení. V případě malého počtu modelů je možné obě položky převést do tabulky číselníků. Protože model a výrobce spolu souvisí, je možné navázat VYROBCE v číselníkách k MODELU.
- CENA značí pořizovací cenu zařízení.
- DATUM_PORIZENI představuje datum, kdy bylo zařízení zakoupeno.
- POPIS je textová položka pro bližší popis zařízení.
- POZNAMKA obsahuje další dodatečné informace k zařízení.

```
CREATE TABLE `other_hw`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `typ` INT NOT NULL ,
 `druh` VARCHAR( 255 ) ,
 `nazev` VARCHAR( 20 ) NOT NULL ,
 `sn` VARCHAR( 55 ) ,
 `ic` VARCHAR( 20 ) NOT NULL ,
 `lokalita` INT NOT NULL ,
 `prechodna_lokalita` INT,
 `velikost` FLOAT,
 `model` VARCHAR( 20 ) ,
 `vyrobce` VARCHAR( 20 ) ,
`cena` FLOAT NOT NULL ,
 `datum_porizeni` DATE NOT NULL ,
 `popis` LONGTEXT,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` ) ,
 UNIQUE (`ic`)
)
TYPE = innodb;
```

Příklad insertu do tabulky ostatního hardwaru[22]

Do tabulky vložíme harddisk. Přes vazební tabulky by následně mohl mít tento harddisk vazbu například na PC, v němž je zapojen. Jedná se zcela běžný insert.

```
INSERT INTO `other_hw` ( `id` , `typ` , `druh` , `nazev` , `sn` , `ic`
, `lokalita` , `prechodna_lokalita` , `velikost` , `model` , `vyrobce`
, `cena` , `datum_porizeni` , `popis` , `poznamka` )
VALUES
(NULL, '0', 'Harddisk', 'Seagate', NULL , 'ic12345', '1', NULL ,
'120', 'ST.BAR.7200', 'Seagate', '1600', '2007-12-19', NULL , NULL)
```

[22] Jelikož se jedná o demonstraci, kdy nemáme k dispozici skutečná data pro číselníky, bude v insertu pod položkou TYP číslo 0. Se skutečnými daty by tam patřila patřičná položka z tabulky číselníků.

3.1.14. Tabulka hw

Tato tabulka obsahuje samotné infrastrukturní prvky, jako jsou servery, osobní počítače, notebooky, switche, nebo huby.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
typ	INTEGER	Not null
nazev	VARCHAR(20)	Not null, Unique
sn	VARCHAR(55)	Null
ic	VARCHAR(20)	Not null, Unique
lokalita	INTEGER	Not null
prechodna_lokalita	INTEGER	Null
model	VARCHAR(20)	Null
vyrobce	VARCHAR(20)	Null
ram	FLOAT	Null
ip	VARCHAR(15)	Not null
mac	VARCHAR(17)	Null
os	VARCHAR(55)	Null
pocet_cpu	INTEGER	Null
typ_cpu	VARCHAR(20)	Null
frekvence_cpu	FLOAT	Null
pocet_portu	INTEGER	Null
speed	INTEGER	Null
ucel_zarizeni	LONGTEXT	Null
cena	FLOAT	Not null
datum_porizeni	DATE	Not null
popis	LONGTEXT	Null
poznamka	LONGTEXT	Null

- ID je primární klíč tabulky.
- TYP je číselná hodnota s vazbou na číselník určující typ zařízení.
- NAZEV je název zařízení, v tabulce `hw` se jedná o unikátní povinnou hodnotu.
- SN je sériové číslo zařízení.
- IC je inventární číslo, které je unikátní v rámci celé databáze.
- LOKALITA a PRECHODNA_LOKALITA mají vazbu na tabulku `lokalita`, která určuje jejich konkrétní umístění.
- MODEL a VYROBCE mohou být v případě menšího počtu modelů přeřazeny jako položky číselníků.
- RAM je velikost ram paměti u zařízení, která ram paměť mají.
- IP a MAC jsou hodnoty ip a mac adres v případě, že je typ zařízení nevylučuje. Na obou položkách by bylo možno vytvořit například funkci, která by kontrolovala validitu dat. U IP adresy například rozložením adresy na části a kontrolou, že žádná není větší než 255. Tuto kontrolu lze ale efektivněji vytvořit již na úrovni PHP.
- OS je operační systém u zařízení, která nějaký operační systém mají. Je možné použít položku číselníku u menšího počtu operačních systémů.
- POČET_CPU, TYP_CPU a FREKVENCE_CPU jsou údaje k procesorům u typu zařízení, který procesory má. Bylo by možné informace o procesorech dát do zvláštní tabulky s vazbou na tabulku `hw`, ale vzhledem k tomu, že se nepředpokládá, že by některé zařízení mělo více než jeden typ procesoru, je tento krok zbytečný.
- POČET_PORTU a SPEED jsou informace o konektivitě zařízení.

- UCEL_ZARIZENI značí primární účel zařízení.
- CENA značí pořizovací cenu zařízení.
- DATUM_PORIZENI je datum kdy bylo zařízení pořízeno.
- POPIS je textový popis zařízení.
- POZNAMKA je textová poznámka obsahující nějaké doplňující informace.

```
CREATE TABLE `hw`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `typ` INT UNSIGNED NOT NULL ,
 `nazev` VARCHAR( 20 ) NOT NULL ,
 `sn` VARCHAR( 55 ) ,
 `ic` VARCHAR( 20 ) NOT NULL ,
 `lokalita` INT NOT NULL ,
 `prechodna_lokalita` INT,
 `model` VARCHAR( 20 ) ,
 `vyrobce` VARCHAR( 20 ) ,
 `ram` FLOAT,
 `ip` VARCHAR( 15 ) NOT NULL ,
 `mac` VARCHAR( 17 ) ,
 `os` VARCHAR( 55 ) ,
 `pocet_cpu` INT,
 `typ_cpu` VARCHAR( 20 ) ,
 `frekvence_cpu` FLOAT,
 `pocet_portu` INT,
 `speed` INT,
 `ucel_zarizeni` LONGTEXT,
 `cena` FLOAT NOT NULL ,
 `datum_porizeni` DATE NOT NULL ,
 `popis` LONGTEXT,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` ) ,
 UNIQUE (`nazev`,`ic`)
)
TYPE = innodb;
```

Příklad insertů do tabulky HW[23]

Do tabulky vložíme dvě zařízení, jeden server a jeden aktivní prvek. Na insertech není nic neobvyklého co by bylo zapotřebí speciálně rozebírat.

```
INSERT INTO `hw` ( `id` , `typ` , `nazev` , `sn` , `ic` , `lokalita` ,
`prechodna_lokalita` , `model` , `vyrobce` , `ram` , `ip` , `mac` ,
`os` , `pocet_cpu` , `typ_cpu` , `frekvence_cpu` , `pocet_portu` ,
`speed` , `ucel_zarizeni` , `cena` , `datum_porizeni` , `popis` ,
`poznamka` )
VALUES
(NULL, '0', 'mx server', 'sn1234', 'ic1234', '1', NULL , 'DL 360 G3',
'HP', '4', '255.255.255.255', '01-23-45-67-89-ab', 'MS server 2003',
'2', 'C2D', '3200', NULL , NULL , 'MX server', '150000', '2007-12-31',
NULL , NULL),

(NULL, '0', 'hub 1.7', 'sn12345', 'ic12345', '1', NULL , 'Cisco 1800',
'Cisco', NULL , '10.10.10.10', 'ff:ff:ff:ff:ff:ff', NULL , NULL , NULL
, NULL , '40', '1000', 'Root hub', '75000', '2007-12-30', NULL , NULL)
```

[23] Jelikož se jedná o demonstraci, kdy nemáme k dispozici skutečná data pro číselníky, bude v insertu pod položkou TYP číslo 0. Se skutečnými daty by tam patřila patřičná položka z tabulky číselníků.

3.1.15. Tabulka hw_disk

Tato tabulka je rozšiřující k tabulce `hw` a obsahuje seznam jednotlivých harddisků zařízení. Vzhledem k tomu, že každé zařízení může mít více harddisků s různými velikostmi, od různých výrobců s různými sériovými čísly, je toho vhodné řešení zajištění vazby hardwaru na harddisky.

Název	Datový typ	Další atributy
id	INTEGER	Primární klíc, Auto increment, Not null
id_hw	INTEGER	Cizí klíč tabulky hw, Not null
typ	VARCHAR(20)	Null
velikost	FLOAT	Null
model	VARCHAR(20)	Null
vyrobce	VARCHAR(20)	Null
zapojeni	VARCHAR(20)	Null
poznamka	LONGTEXT	Null

- ID je primární klíc tabulky
- ID_HW je cizí klíč z tabulky `hw`, který tvoří vazbu mezi zařízením a harddiskem.
- TYP je typ harddisku.
- VELIKOST je velikost harddisku, vzhledem k velikostem dnešních harddisků je vhodné použít jako jednotku GB.
- MODEL je model harddisku
- VYROBCE je výrobce harddisku.
- ZAPOJENI označuje jakým způsobem jsou harddisky zapojeny.
- POZNAMKA obsahuje další dodatečné informace k jednotlivým harddiskům.

```
CREATE TABLE `hw_disk`
(`id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
 `id_hw` INT UNSIGNED NOT NULL ,
 `typ` VARCHAR(20 ) ,
 `velikost` FLOAT,
 `model` VARCHAR( 20 ) ,
 `vyrobce` VARCHAR( 20 ) ,
 `zapojeni` VARCHAR( 20 ) ,
 `poznamka` LONGTEXT,
 PRIMARY KEY ( `id` ),
 CONSTRAINT `hw_disk` FOREIGN KEY `hw_disk` ( `id_hw` ) REFERENCES
`hw` ( `id` ) ON DELETE RESTRICT ON UPDATE RESTRICT
)
TYPE = innodb;
```

Příklad insertu do tabulky harddisků

Do tabulky vložíme tři harddisky, dva stejné velikosti zapojené do RAID1 a jeden samostatný. Samotný insert opět neskýtá žádná úskalí, kromě nutnosti ohlídat si hodnotu ID_HW, která je cizím klíčem z tabulky `hw`.

```
INSERT INTO `hw_disk` ( `id` , `id_hw` , `typ` , `velikost` , `model`
, `vyrobce` , `zapojeni` , `poznamka` )
VALUES
(NULL, '1', 'SSD HDD', '100', 'SSD', 'Samsung', 'RAID1', NULL),
(NULL, '1', 'SSD HDD', '100', 'SSD', 'Samsung', 'RAID1', NULL),
(NULL, '1', 'Western Digital 15K', '60', 'WD8276.15K', 'Western
Digital', 'SATA', 'swap disk')
```

3.2. Rady pro vkládání dat do tabulek

Před vkládáním dat do databáze je dobré dát si pozor zejména

I. Zda se v tabulce nevyskytují cizí klíče.

II. Pokud se do tabulky vkládají hodnoty z číselníků nebo lokalit, je vhodné zkontrolovat, zda zadávaná položka číselníků případně lokalit skutečně existuje.

III. Dbát zvýšené pozornosti na datové typy jednotlivých entit v tabulce.

IV. Složené klíče/indexy tabulek.

4. Závěr

4.1. Shrnutí

Jak jsme si v této bakalářské práci ukázali, současný stav evidence infrastruktury a majetku má svá slabá místa a pár nedostatků, které práci s evidencí samotnou neusnadňují, ba naopak v některých ohledech omezují a ztěžují. Ze zadání bakalářské práce je evidentní, že potřeba ucelené, jednoduché a hlavně přehledné evidence je více než žádoucí a pouze potvrzuje fakt, že současný stav evidence již není zcela vyhovující.

Z práce je také zřejmé, jaké výhody skýtá možnost evidence infrastruktury formou databáze oproti starším metodám evidence, jako je využívání jednoduchých tabulkových procesorů nebo obyčejných textových dokumentů či ještě hůře evidence v čistě papírové podobě.

Dále jsme si ukázali, jaké cesty vedou k vytyčenému cíli, jaká jsou jejich úskalí, slabé či naopak silné stránky, v čem se od sebe různé varianty liší a v čem si mohou být naopak podobné. Zjistili jsme, že získání řešení nemusí být jen v teoretické rovině, ale že se může přesunout i na pole praktického řešení.

V průběhu mé práce jsme postupně zjistili, k jakému účelu lze použít logování, jaké výhody má použití databázového enginu s podporou transakcí, jaké můžeme použít databázové platformy, jaký je smysl číselníků a jaké varianty číselníků máme, jaké položky je vhodné do číselníků zařadit, jaké databázové tabulky jsou výhodné v případě tvorby jiného uživatelského prostředí, než je administrační rozhraní samotné databáze, zjistili jsme, k čemu může být výhodné vést si deníky a seznamy požadavků, jaké tabulky lze spojit do jedné větší, nebo naopak rozdělit na určitý počet menších, kde lze použít k provedení akcí na úrovni databáze triggery a funkce, nebo na co si máme dát pozor při vkládání dat do tabulek.

Teoretické znalosti nabyté studiem, které jsem spojil s praktickými zkušenostmi, získanými prací v oboru, jsem vytvořil práci, která nám poskytla ucelený náhled na postup tvorby a návrhu databáze, provedla nás úskalím prvotní analýzy a ukázala nám, že k cíli vede vždy několik cest.

Na základě mé bakalářské práce by nyní neměl být problém pochopit funkčnost mnou navrhované databáze, její klady i zápory nebo silné i slabé stránky. Nyní bychom již měli být schopni databázi podle návodu vytvořit, nebo rozšířit o další tabulky či položky v tabulkách, zajistit správné vkládání dat do jednotlivých tabulek i práci s nimi, pochopit vzájemné vazby i účel jednotlivých tabulek databáze a vyvarovat se zbytečným chybám při práci jako jsou například cizí klíče nebo položky číselníků.

Vytýčeným cílem mé práce bylo provést analýzu současného stavu evidence infrastruktury na Vyšší odborné škole informačních služeb v Praze a navrhnout databázové řešení pro evidenci infrastruktury a majetku, které by odpovídalo zadaným požadavkům a zároveň patřilo mezi uživatelsky přívětivá řešení.

4.2. Terminologický slovník

Databáze

- Databáze jak jí definuje česká wikipedie je

  „určitá uspořádaná množina informací (dat) uložená na paměťovém médiu. V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim.“[24]
- Tato definice je jistě postačující, avšak rád bych jí doplnil o význam uvedený v American Heritage Dictionary, který uvádí, že databáze je

  „A collection of data arranged for ease and speed of search and retrieval. Also called data bank.“[25]
- Je patrné, že pravá podstata slova databáze vnikne spojením těchto dvou definic. Tudíž že se jedná o množinu informací uložených na paměťovém médiu, určených k jejich prohledávání, archivaci a třídění.

[24] Databáze. (podst. jméno). *Otevřená encyklopedie Wikipedie,* dostupná na adrese [http://cs.wikipedia.org/wiki/Datab%C3%A1ze]

[25] database. (n.d.). *The American Heritage® Dictionary of the English Language, Fourth Edition*. Retrieved December 27, 2007, from Dictionary.com website: http://dictionary.reference.com/browse/database

Data

- Dále považuji za vhodné definovat pojem data. Tímto pojmem je nejčastěji myšlena jakákoli množina údajů, které pro nás mohou a nemusejí mít jakýkoli smysl a význam. Encyklopedie wikipedie[26] definuje pojem data jako
    a. *vyjádření skutečností formálním způsobem tak, aby je bylo možno přenášet nebo zpracovat*
    b. *číselné nebo jiné symbolicky vyjádřené (reprezentované) údaje a hodnoty nějakých entit nebo událostí*
    c. *jakékoli fyzicky (materiálně) zaznamenané znalosti (vědomosti), poznatky, zkušenosti nebo výsledky pozorování procesů, projevů, činností a prvků reálného světa (reality)*
    d. *surovina, z níž se tvoří informace*

Informace

- Informace jsou data, která pro nás mají určitý smysl nebo význam.
- Wordnet Princeton University[27] definuje informaci takto:

    1. a message received and understood

    2. knowledge acquired through study or experience or instruction

    3. formal accusation of a crime

    4. a collection of facts from which conclusions may be drawn; "statistical data"

    5. (communication theory) a numerical measure of the uncertainty of an outcome; "the signal contained thousands of bits of information"

Databázová platforma

- Jedná se programové produkty firem jako Oracle, Microsoft, IBM, MySQL AB,…. Často nesprávně označované také jako databázové servery.

Databázový model

- jedná se v podstatě o způsob ukládání dat a vazeb mezi nimi v databázi. Databázové modely jsou[28]:
    a. *Hierarchická databáze*
    b. *Síťová databáze*
    c. *Relační databáze*
    d. *Objektová databáze*
    e. *Objektově relační databáze*

---

[26] Data. (podst. jméno). *Otevřená encyklopedie Wikipedie,* dostupná na adrese [http://cs.wikipedia.org/wiki/Data]

[27] "information." *WordNet® 3.0*. Princeton University. 30 Dec. 2007. <Dictionary.com http://dictionary.reference.com/browse/information>.

[28] *Wikipedie: Otevřená encyklopedie: Databáze* [online]. c2007 [citováno 30. 12. 2007]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Datab%C3%A1ze&oldid=2046185>

Tabulka

„je jedním ze základních databázových objektů, který slouží k přímému uložení dat do paměťového prostoru relační databáze.“[29]

- Lze si jí snadno představit např. jako dvourozměrnou tabulku se sloupci a řádky. Data se nejčastěji ukládají po řádcích (tj. co řádek to jeden záznam), ale v některých případech jsou data ukládána i po sloupcích.

Primární klíč

„je pole nebo kombinace polí, jednoznačně identifikující každý záznam v databázové tabulce. Žádné pole, které je součástí primárního klíče, nesmí obsahovat hodnotu NULL. Každá tabulka má mít definovaný právě jeden primární klíč (entitní integrita).“[30]

Integritní omezení

- Pravidla pro zajištění správnosti a konzistence uložených dat.
  a. Doménová – zajišťují, aby každá hodnota byla v souladu s přípustnými hodnotami
  b. Entitní – jednoznačná identifikace každého řádku
  c. Referenční – hodnoty cizích klíčů nemohou nabývat hodnot jiných než v odkazovaném primárním klíči

Normalizace databáze

- Technika datové analýzy, která zkoumá závislosti mezi položkami záznamu.
- Cílem je eliminace redundance dat a přizpůsobení databáze co nejvíce k obrazu reality
- Existuje pět úrovní normalizace.

[29] *Wikipedie: Otevřená encyklopedie: Tabulka (databáze)* [online]. c2007 [citováno 30. 12. 2007]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Tabulka_%28datab%C3%A1ze%29&oldid=1949104>

[30] *Wikipedie: Otevřená encyklopedie: Primární klíč* [online]. c2007 [citováno 30. 12. 2007]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Prim%C3%A1rn%C3%AD_kl%C3%AD%C4%8D&oldid=1823866>

4.3. Použité a citované zdroje

[1]. *Wikipedie: Otevřená encyklopedie: Hlavní strana* [online]. c2007 [citováno 30. 12. 2007]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Hlavn%C3%AD_strana&oldid=1869941>

[2]. MySQL AB. *Mysql Reference Manual* [online]. MySQL AB, 2007 , 2007-12-28 (revision: 9455) [cit. 2007-12-30]. Manuál k MySQ\"L. English. Dostupný z WWW: <http://dev.mysql.com/doc/refman/5.1/en/index.html>.

[3]. *Dictionary.com* [online]. Publishing Group, LLC, c2007 [cit. 2007-12-29]. Systém pro vyhledávání v několika výkladových slovnících. English. Dostupný z WWW: <http://dictionary.com/>. http://dictionary.reference.com/.

[4]. *ITIL Official website* [online]. APM Group, c2007 , 18/12/2007 [cit. 2007-12-31]. Oficiální stránky ITIL. English. Dostupný z WWW: <http://www.itil-officialsite.com/home/home.asp>.

[5]. Free Software Foundation,. *GNU Operating System* [online]. c2007 , 007/12/30 18:17:28 [cit. 2007-12-29]. Oficiální stránky projektu GNU. Dostupný z WWW: <http://www.gnu.org/>.

[6]. *PHP official sites* [online]. The PHP Groupe, c2007 , Sun Dec 30 12:06:47 2007 PST [cit. 2007-12-30]. Oficiální stránky projektu PHP. Dostupný z WWW: <http://php.net/>. Http://cz2.php.net/.

[7]. phpMyAdmin Devel Team. *PhpMyAdmin official sites* [online]. [2007] , 2007-08-21 [cit. 2007-12-30]. Oficiální stránky projektu phpMyAdmin. English. Dostupný z WWW: <http://www.phpmyadmin.net/home_page/index.php>. http://wiki.cihar.com/pma/Welcome_to_phpMyAdmin_Wiki.

[8]. ŠIMůNEK, Milan. *SQL : Kompletní kapesní průvodce*. 1999. vyd. [s.l.] : Grada Publishing a.s., 1999. 248 s. Průvodce. ISBN 80-7169-692-7.

[9]. Firebird Foundation Incorporated. *Firebird SQL project official site* [online]. c2007 [cit. 2007-12-30]. Oficiální stránky projektu Firebird SQL. English. Dostupný z WWW: <http://firebirdsql.org/>.

[10]. PostgreSQL Global Development Group. *PostgreSQL official sites* [online]. c2007 [cit. 2007-12-31]. Oficiální stránky projektu PostgreSQL. Dostupný z WWW: <http://www.postgresql.org/>.

[11]. KUČEROVÁ, Helena. *Databázové systémy : Sylaby ke kurzu*. Praha : Vyšší odborná škola informačních služeb, 2004. 110 s. PDF verze ke stažení na adrese: <http://info.sks.cz/users/ku/DOKUMENTY/das_syl.pdf>

4.4. Doplňující a rozšiřující literatura

4.4.1. Analýza a návrh databází

[12]. BUCHALCEVOVÁ, Alena. Metodiky vývoje a údržby informačních systémů: Kategorizace, agilní metodiky, vzory pro návrh metodiky. Praha : Grada Publishing a.s., 2005. 163 s. ISBN 80-247-1075-7

[13]. HERNANDEZ, Michael J. *Návrh databází*. Praha : Grada Publishing a.s., 2005. ISBN 80-247-0900-7

[14]. POKORNÝ, Jaroslav. *Konstrukce databázových systémů.* Dotisk 1. vyd. Praha : ČVUT, 2001. 166 s. ISBN 80-01-01935-7

[15]. VRANA, Ivan a RICHTA, Karel. Zásady a postupy zavádění podnikových informačních systémů: Praktická příručka pro podnikové manažery. 1. vyd. Praha: Grada Publishing a.s., 2005. 187 s.
ISBN 80-247-1103-6

[16]. RIORDAN, Rebecca M. *Vytváříme relační databázové aplikace*. Praha: Computer Press, 2000. 280 s. ISBN 80-7226-360-9

[17]. OPPEL, Andy. *Databáze bez předchozích znalostí*. 2006. vyd. [s.l.] : Computer Press a.s., 2006. 320 s. Databáze. ISBN 80-251-1199-7.

4.4.2. Dotazovací jazyky

[18]. POKORNÝ, Jan. Uíme se SQL. Praha : Plus, 1993. 566 s.

[19]. POKORNÝ, Jaroslav. Dotazovací jazyky. Praha : Karolinum, 2002. 255 s. ISBN 80-246-0497-3

[20]. GROFF, James R., WEINBERG, Paul N. SQL : Kompletní průvodce. 2005. vyd. [s.l.] : Computer Press a.s., 2005. 936 s. Databáze. ISBN 80-251-0369-2.

[21]. LACKO, Luboslav. *SQL : Hotová řešení*. 2003. vyd. [s.l.] : Computer Press a.s., 2003. 276 s. Databáze. ISBN 80-7226-975-5.

[22]. HERNANDEZ, Michael J., VIESCAS, John L. Myslíme v jazyku SQL : Tvorba dotazů. 2004. vyd. [s.l.] : Grada Publishing a.s., 2004. 380 s. Programování. ISBN 80-247-0899-X.

4.4.3. Databázové platformy

[23]. KRUCZEK, Aleš. *Microsoft Office Access 2007 : Podrobná uživatelská příručka*. 2007. vyd. [s.l.] : Computer Press a.s., 2007. 365 s. ISBN 978-80-251-1608-1.

[24]. LACKO, Luboš. Oracle : Správa, programování a použití databázového systému. 2007. vyd. [s.l.] : Computer Press a.s., 2007. 576 s. ISBN 978-80-251-1490-2.

[25]. STANEK, William R. *Microsoft SQL Server 2005 : Kapesní rádce administrátora*. 2007. vyd. [s.l.] : Computer Press a.s., 2007. 544 s. ISBN 80-251-1211-X.

[26]. WELLING, Luke, THOMSON, Laura. *MySQL : Průvodce základy databázového systému*. 2005. vyd. [s.l.] : Computer Press a.s., 2005. 256 s. ISBN 80-251-0671-3.

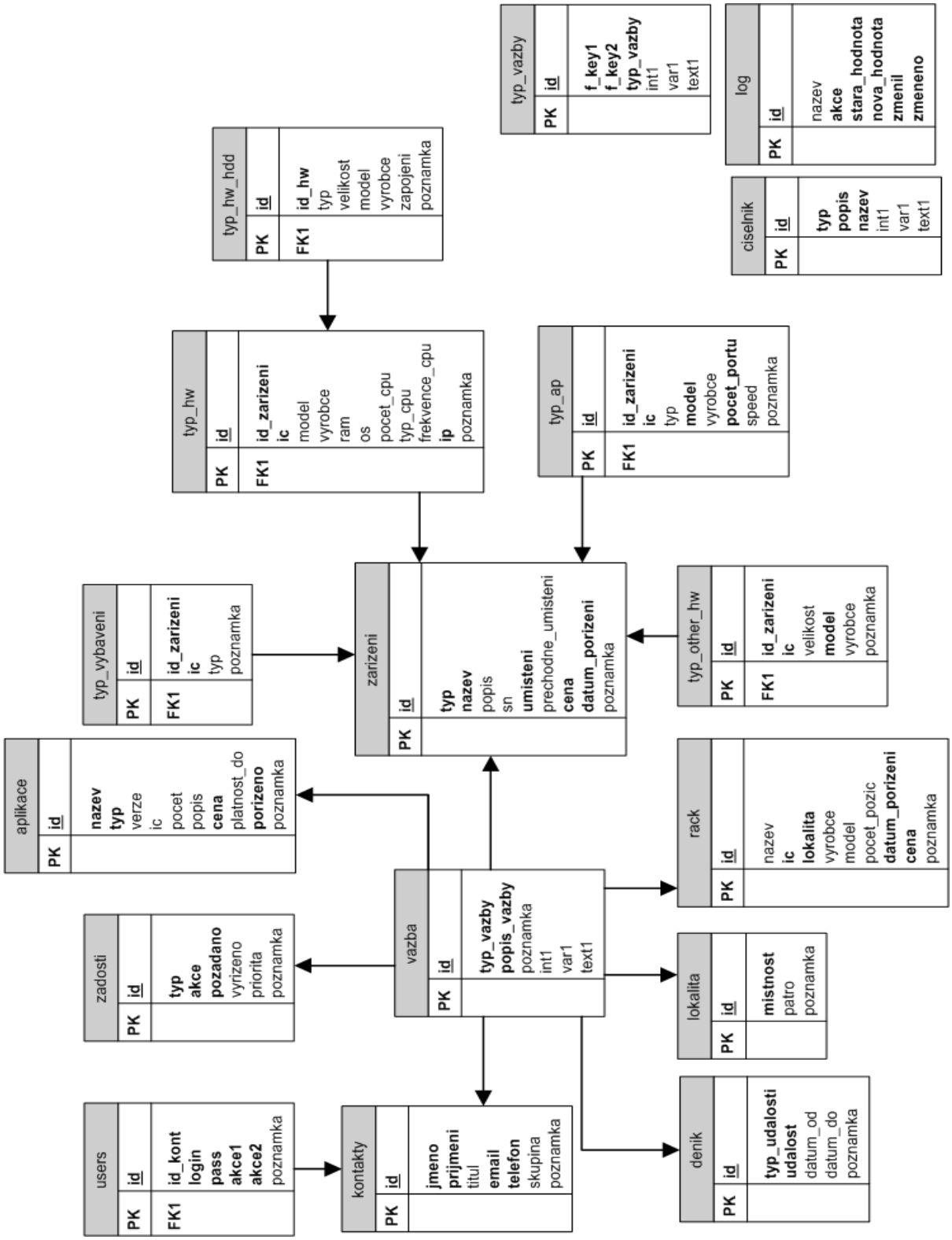
[27]. MOMJIAN, Bruce. *PostgreSQL : Praktický průvodce*. 2003. vyd. [s.l.] : Computer Press a.s., 2003. 424 s. ISBN 80-722-6954-2.

[28]. CÍSAŘ, Pavel. *InterBase/FireBird : Tvorba, administrace a programování databáz*. 2003. vyd. [s.l.] : Computer Press a.s., 2003. 464| s. ISBN 80-7226-946-1.

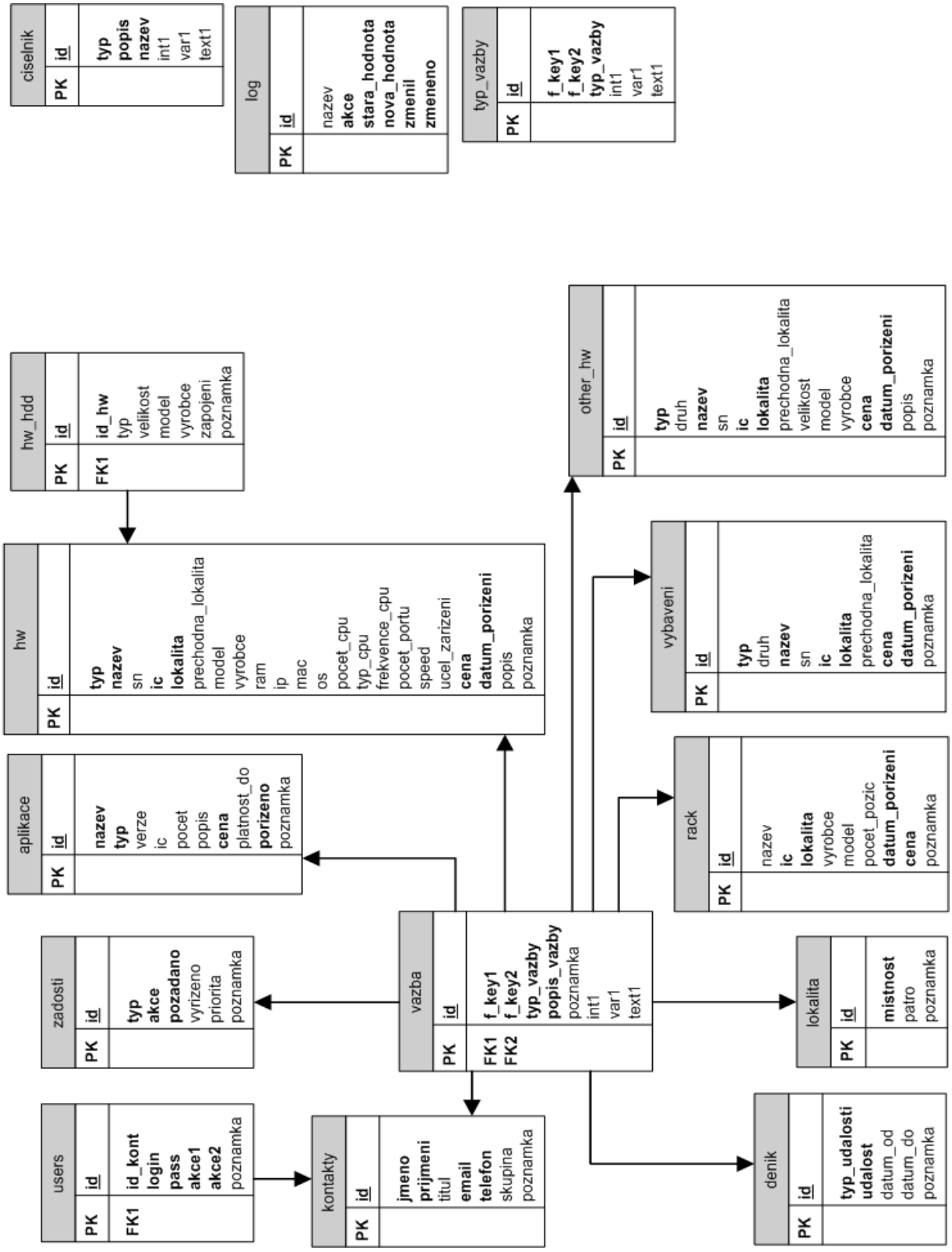
[29]. KOCAN, Marek (šéfredaktor). *Databázový svět : Informační portál ze světa databázových technologií* [online]. AVRE Publishing, spol. s r.o. , c2004 [cit. 2007-12-31]. Informace z databázového světa. Český. Dostupný z WWW: <http://www.dbsvet.cz/>. ISSN 1213-5933.

4.5. Přílohy

4.5.1. Schéma 1



4.5.2. Schéma 2


ciselnik
PK id
typ
popis
nazev
int1
var1
text1
log
PK id
nazev
akce
stara_hodnota
nova_hodnota
zmenil
zmeneno
typ_vazby
PK id
f_key1
f_key2
typ_vazby
int1
var1
text1
hw_hdd
PK id
FK1 id_hw
typ
velikost
model
vyrobce
zapojeni
poznamka
hw
PK id
typ
nazev
sn
ic
lokalita
prechodna_lokalita
model
vyrobce
ram
ip
mac
os
pocet_cpu
typ_cpu
frekvence_cpu
pocet_portu
speed
ucel_zarizeni
cena
datum_porizeni
popis
poznamka
other_hw
PK id
typ
druh
nazev
sn
ic
lokalita
prechodna_lokalita
velikost
model
vyrobce
cena
datum_porizeni
popis
poznamka
vybaveni
PK id
typ
druh
nazev
sn
ic
lokalita
prechodna_lokalita
cena
datum_porizeni
poznamka
aplikace
PK id
nazev
typ
verze
ic
pocet
popis
cena
platnost_do
porizeno
poznamka
rack
PK id
nazev
ic
lokalita
vyrobce
model
pocet_pozic
datum_porizeni
cena
poznamka
zadosti
PK id
typ
akce
pozadano
vyrizeno
priorita
poznamka
vazba
PK id
FK1 f_key1
FK2 f_key2
typ_vazby
popis_vazby
poznamka
int1
var1
text1
lokalita
PK id
mistnost
patro
poznamka
users
PK id
FK1 id_kont
login
pass
akce1
akce2
poznamka
kontakty
PK id
jmeno
prijmeni
titul
email
telefon
skupina
poznamka
denik
PK id
typ_udalosti
udalost
datum_od
datum_do
poznamka