VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH DATABÁZE SKLADOVÉHO HOSPODÁŘSTVÍ

DESIGN OF DATABASE OF WAREHOUSE MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — RADIM SEKULA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN LUHAN, Ph.D.
SUPERVISOR

BRNO 2015

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2014/2015
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Sekula Radim

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 skladového hospodářství

v anglickém jazyce:

Design of Database of Warehouse Management

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:

CONOLLY T., C. BEGG a R. HOLOWCZAK. Mistrovství - Databáze: Profesionální průvodce tvorbou efektivních databází. 1. vyd. Praha: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.
ELMASRI, R. and S. B. NAVATHE. Fundamentals of Database Systems. 6th ed. Boston: Addison Wesley, 2010. 1172 p. ISBN 978-0-136-08620-8.
LACKO, L. 1001 tipů a triků pro SQL. 1. vyd. Brno: Computer Press, 2011. 416 s. ISBN 978-80-251-3010-0.
OPPEL, A. Databáze bez předchozích znalostí. 1. vyd. Brno: Computer Press, 2006. 316 s. ISBN 80-251-1199-7.
STEPHENS, R. K. a R. R. PLEW. Naučte se SQL za 21 dní. 1. vyd. Brno, Computer Press 2004. 491 s. ISBN 80-722-6870-8.

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

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2014/2015.

L.S.

doc. RNDr. Bedřich Půža, CSc.	doc. Ing. et Ing. Stanislav Škapa, Ph.D.
Ředitel ústavu	Děkan fakulty

V Brně, dne 28.2.2015

Abstrakt

Bakalářská práce se zaměřuje na návrh databáze skladového hospodářství ve firmě DATEL PLUS s.r.o. Práce je rozdělena na teoretickou část a praktickou část. V teoretické části jsou uvedeny potřebné informace a teorie nezbytná k úspěšnému splnění cíle práce. V praktické části je analyzována současná situace firmy a je navrženo vlastní řešení daného problému.

Abstract

The bachelor thesis focuses on the design of the database of stock management in the company DATEL PLUS s.r.o. The work is divided into a theoretical part and a practical part. The theoretical part provides the necessary information and theories important for successful fulfilling the goal of the work. The practical part analyses the current situation of the company and offers a solution to the problem mentioned.

Klíčová slova

Databáze, databázový systém, datové modelování, relační datový model, SQL

Key words

Database, database system, data modeling, relational data model, SQL

Bibliografická citace

SEKULA, R. *Návrh databáze skladového hospodářství*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2015. 63 s. Vedoucí bakalářské práce Ing. Jan Luhan, Ph.D.

Čestné prohlášení

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

V Brně dne 27. května 2015 ……

Radim Sekula

Poděkování

Rád bych tímto poděkoval vedoucímu práce panu Ing. Janu Luhanovi, Ph.D. za vstřícný přístup, cenné rady a připomínky, které mi pomohly při řešení této bakalářské práce.

Dále bych také rád poděkoval společnosti DATEL PLUS s.r.o. za poskytnuté informace, podklady a konzultace během zpracovávání této práce.

Obsah

ÚVOD

V současné době jsme obklopeni nebývalým množstvím různých komunikačních prostředků, které nám umožňují sdílení informací v rozlišných sítích. Informace jsou shromažďovány na serverech ve formě dat a ty představují místo pro využití databázových systémů. Ať chceme nebo ne, tak každý z nás přichází prakticky denně do styku s databázemi, například ve škole, zaměstnání, obchodech, bankách či úřadech. Druh uchovávaných informací se specificky odvíjí od toho, na co se konkrétní společnost specializuje. Ať už jde o velké, střední nebo malé podniky, tak každý z nich má potřebu nějaké informace uchovávat. Databáze zde nabízí spoustu možností a to především v tom, jak zlepšit a zefektivnit uchovávání informací či jak zjednodušit provádění různých operací nad ukládanými daty.

Tato bakalářská práce se bude zabývat teoretickým pozadím databází, návrhem a implementací databáze, která bude sloužit k evidenci skladového hospodářství v konkrétním podniku.

1 CÍL PRÁCE A METODIKA

Cílem této bakalářské práce je vytvořit návrh databáze skladového hospodářství ve společnosti DATEL PLUS s.r.o.

Dílčími kroky k dosažení tohoto cíle jsou počáteční analýza, vypracování návrhu databáze, spočívající ve vytvoření konceptuálního, logického a fyzického datového modelu, a ověření funkčnosti databáze testovacími daty.

Analýza bude spočívat především v konzultacích s IT specialistou společnosti, díky kterým budou stanoveny požadavky na funkcionalitu databáze. Metodou pro vypracování návrhu databáze je nejprve určení notace a nástroje pro tvorbu datového modelu a následně taktéž určení nástroje pro implementaci databáze.

Hlavní náplní této práce je návrh relačního datového modelu, který bude korespondovat s reálnou situací a potřebami firmy.

2 TEORETICKÁ VÝCHODISKA PRÁCE

Cílem této části je poskytnout základní přehled o databázích a metodologii návrhu databáze jako takové. Tyto poznatky budou tvořit základ pro celkovou tvorbu této bakalářské práce.

2.1 Základní pojmy

Před samotnou teorií návrhu databáze je nezbytné představit některé důležité základní pojmy, jako jsou databáze, databázový systém nebo databázový model.

2.1.1 Databáze

„Databáze je kolekce vzájemně souvisejících datových položek, které jsou spravovány jako jediná jednotka." [7, str. 9]

Je to soubor dat používaný k modelování některých typů organizačních struktur nebo organizačních procesů. Vůbec nezáleží na tom, jestli ke shromažďování a ukládání dat používáte papír nebo počítačový program. Dokud shromažďujete data jakýmkoli organizovaným způsobem, máte databázi. [9]

Databází tak rozumíme místo, kde jsou o reálném světě uchovávána data. A to takovým způsobem, abychom s nimi mohli snadně pracovat, mazat nebo přidávat další. Příkladem databáze je kartotéka knihovny, v které se uchovávají jména jednotlivých autorů a jejich knih. Zde jde ovšem o databázi papírovou, jež je v současnosti již zastaralým způsobem uchovávání informací. V dnešní době jsou využívány převážně relační databáze. Pojmenovaná datová struktura, uložená v databázi, se označuje jako databázový objekt. V závislosti na použití určitého databázového modelu a databázového systému se odlišují konkrétní podporované typy těchto objektů. [7]

2.1.2 Databázový systém

„Databázový systém (Database Management Systems neboli DBMS) je softwarový systém, který uživateli umožňuje definovat, vytvářet a udržovat databázi a poskytuje řízený přístup k této databázi.“ [2, str. 38]

DBMS usnadňuje procesy *definování*, *konstrukce*, *manipulace* a *sdílení* databází mezi různými uživateli či aplikacemi. **Definování** databáze zahrnuje specifické datové

typy, struktury a omezení údajů, které mají být v databázi ukládány. V DBMS jsou taktéž uloženy definice databáze či popisné informace ve formě datového slovníku (katalogu) – tzv. meta-data. **Konstrukce** databáze představuje proces ukládání dat na nějaké paměťové médium, které je pomocí DBMS řízeno. **Manipulace** databáze obsahuje funkce jako je dotazování k získání konkrétních údajů, aktualizace k odrážení změn v reálném světě a generování sestav (reportů) z uložených dat. **Sdílení** databáze umožňuje současný přístup více uživatelů či programů k databázi. [5]

DBMS rozumíme tedy software, který je dodáván příslušným výrobcem databází. Softwarové produkty jako Microsoft Access, Oracle, Microsoft SQL Server, Sybase, DB2, INGRES a MySQL jsou tedy příkladem databázových systémů. V české literatuře bývá databázový systém někdy označovaný jako Systém řízení báze dat (SŘBD). [6]

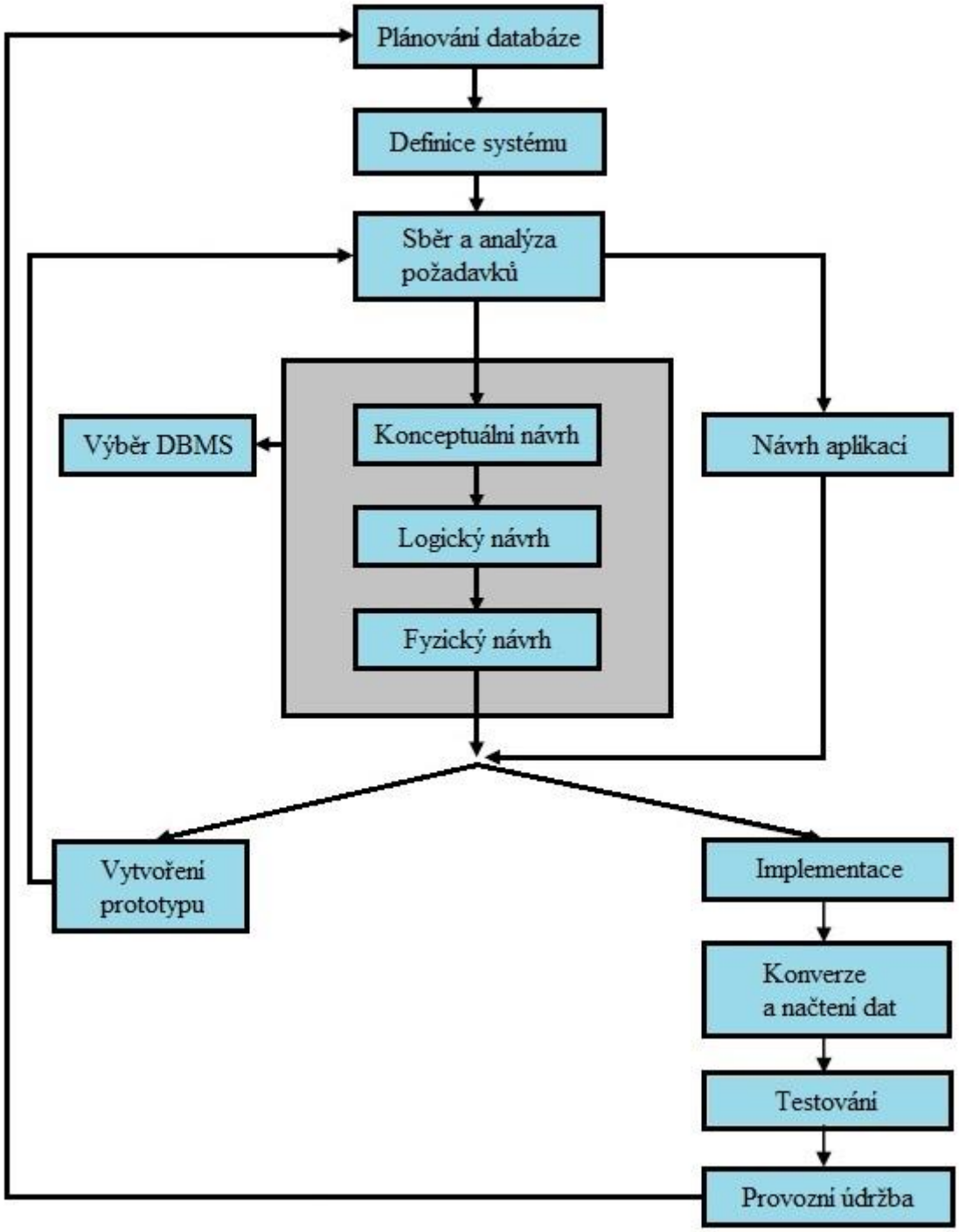
2.1.3 Databázový model

„Databázový model vyjadřuje způsob uspořádání dat v databázi, která tak odráží podobu reálného světa.“ [6, str. 17]

Databázovým modelem označujeme v podstatě architekturu, dle které jsou objekty databázovým systémem ukládány do databáze a která tyto objekty navzájem spojuje. Mezi nejrozšířenější modely databází patří hierarchický, síťový, relační, objektově orientovaný či objektově relační model. [6]

2.2 Životní cyklus databáze

Při vývoji databázového systému je potřeba použít určité techniky a nástroje, které popisuje uspořádaný seznam fází. Tento seznam fází představuje životní cyklus vývoje databázového systému. Typicky tvoří plánování databáze, definice systému, sběr a analýzu požadavků, návrh databáze, (volitelně) výběr DBMS, návrh aplikací, (volitelně) vytvoření prototypu, implementace, konverze a načtení dat, testování a provozní údržbu. Jednotlivé fáze tohoto cyklu ovšem nenásledují striktně po sobě, ale můžou být opakovány na základě zpětné vazby. Např. při problémech během návrhu databáze je někdy nutné provést dodatečné kroky při sběru a analýze požadavků. Obrázek č. 1 popisuje životní cyklus vývoje databázového systému pro relační DBMS. [2]



Obrázek č. 1: Životní cyklus databáze (2, str. 110)

2.3 Návrh databáze

„Návrh databáze je proces vytvoření návrhu, který bude podporovat celkové poslání a dílčí cíle požadovaného databázového systému organizace.“ [2, str. 115]

Během návrhu databáze je určena struktura databáze. Při vytváření systému splňujícího potřeby organizace, je kladen důraz na přístup řízený daty. Tento přístup spočívá v tom, že se prvotně myslí na data a teprve až druhotně na aplikace. Návrh databáze je klíčový pro přijetí systému koncovým uživatelem. Tvoří ho tři hlavní fáze nazývané konceptuální, logický a fyzický návrh. [2]

2.3.1 Konceptuální návrh

Konceptuální návrh databáze je procesem vytváření konceptuálního modelu dat užívaných v prostředí bez jakýchkoli úvah týkající se fyzické implementace či jiných podrobností, jako například výběr cílového databázového systému, programovacího jazyku nebo hardwarové platformy.

Náplní fáze konceptuálního návrhu je identifikace důležitých objektů, které je v databázi nezbytné reprezentovat, a vztahy mezi těmito objekty. [2]

2.3.2 Logický návrh

Nejdůležitější částí návrhu je konzistence, integrita a přesnost dat. To je zaručeno při úspěšném vytvoření logického modelu. Ten je stále nezávislý na platformě, avšak oproti konceptuálnímu modelu jsou již vytvořeny tabulky s příslušnými atributy a vazební entity určené vztahy. Cílem je tedy převést abstraktní konceptuální návrh na model obsahující přesně definované entity, atributy a vazby. V této fázi je též provedena normalizace databáze. [9]

Výsledkem fáze logického návrhu je logický model, který je již založen na specifickém modelu dat (např. relační datový model), ale stále nezávislý na konkrétním DBMS či jiných implementačních detailech. [2]

2.3.3 Fyzický návrh

„Fyzický návrh databáze je proces vytvoření popisu implementace databáze na vnějších paměťových zařízeních; popisuje podkladové tabulky, organizaci souborů, indexy použité kvůli efektivnímu přístupu k datům a všechna související integritní a bezpečnostní omezení.“ [2, str. 258]

Poslední fáze návrhu databáze představuje převod logického návrhu na fyzický model. Při fyzickém návrhu databáze rozhodujeme, jakým způsobem implementovat logický návrh do cílového DBMS, na němž je velmi závislá velká část fyzického

návrhu. U logického návrhu je pokládána otázka *co*?, kdežto při fyzickém návrhu se zabýváme otázkou *jak?*. [2]

2.4 Relační datový model

V současnosti je relační model, založený na teorií relací, nejvíce rozšířenou a používanou metodou. Vzniká spojením několika lineárních modelů prostřednictvím relačních klíčů. Data jsou ukládána ve formě dvourozměrných tabulek. Jednoznačným primárním klíčem jsou identifikovány jednotlivé věty tabulek. Spojení mezi tabulkami není trvalé, ale pouze po dobu potřebnou k získání dat. V relačním modelu je možné zachycení nejen dat o požadovaných objektech, ale i jejich vzájemných vztahů. [3]

2.4.1 Relační databáze

Relační databáze je databáze založená na relačním modelu. Fyzickou reprezentací dat jsou normalizované dvourozměrné tabulky, které lze spojovat dle potřeby. Kombinace (spojení) tabulek umožňuje vytvářet pohledy, zobrazující se taktéž jako dvourozměrné tabulky. Tato schopnost nezávislého využití tabulek či spojení s ostatními tabulkami, bez jakékoliv předem definované hierarchii nebo posloupnosti, činí relační databázi velmi flexibilní. [7]

2.4.2 E-R model

„Modelování entit a relací je proces vizuálního vyjádření entit, atributů a relací do podoby tzv. diagramu entit a relací neboli ER diagramu.“ [6, str. 179]

ER se používá z důvodu odlišného způsobu vnímání dat návrháři, programátory a koncovými uživateli. Aby návrh databáze odpovídal požadavkům zadavatelů, je důležitý celkový společný pohled. Tento diagram neobsahuje víceznačnosti a není technický, tudíž je dobře čitelný i pro neodborné pracovníky a odborníkům z řad IT poskytuje velice cenné informace. [2]

2.4.3 Entita

Entitou rozumíme množinu reálných objektů se stejnými vlastnostmi. Každou entitu lze identifikovat podle jména a jedinečných vlastností. [2]

2.4.4 Atribut

Atribut je určitá charakteristika nebo vlastnost entity, která pomáhá identifikovat entitu. [2]

2.4.5 Tabulka

Tabulka je dvourozměrná struktura, která se skládá z řádků a sloupců. Každá tabulka představuje entitu, která má být v databázi znázorněna, a každý řádek zobrazuje jeden výskyt dané entity. [7]

2.4.6 Relace

Souvislosti mezi tabulkami relační databáze reprezentuje relace. Prostřednictvím relací je možné provázání tabulek formálním způsobem, díky kterému je snadné spojit data z více tabulek, ale flexibilně přitom zahrnout pouze informace, které potřebujete. [1]

2.4.7 Klíče

- **Primární klíč** - jde o jednoznačný identifikátor každého záznamu v tabulce. Jednoznačný je tím, že v tabulce neexistuje věta, která by měla stejnou hodnotu. Taktéž musí být minimální a musí obsahovat konkrétní hodnotu, takže nesmí nabývat hodnoty NULL (prázdné hodnoty). [4]
- **Kandidátní klíč** – klíč stejných vlastností jako primární klíč, ale není určen jako primární klíč. Kandidátních klíčů může být více, a pokud nejde o primární klíč, tak se ostatní nazývají jako alternativní. [3]
- **Cizí klíč** – nějaký klíč v tabulce (obvykle se jedná o jedno pole), který odkazuje na jedno či více polí v jiné tabulce [1]

2.4.8 Reprezentace vztahů

Pomocí vztahů je možné vzájemně sdružovat entity. V E-R modelu jsou obsaženy **třídy vztahů**, které představují asociace mezi třídami entit, a **instance vztahů**, což jsou asociace mezi instancemi entit.

Třída vztahů se může týkat více tříd entit. Jako **stupeň vztahu** je označován počet tříd entit ve vztahu. Nejčastěji používanými vztahy jsou vztahy druhého stupně, označované jako **binární vztahy**. Vztahům třetího stupně říkáme **ternární vztahy**. [8]

Tři typy binárních vztahů:

- **Vztah 1:1** (jedna k jedné) - instanci jedné entity můžeme přiřadit k maximálně jedné instanci jiné entity a naopak.
- **Vztah 1:N** (jedna k více) - nejběžnější typ relace, kdy je libovolnou instanci první entity možné přiřadit k jedné nebo více instancím druhé entity, ale naopak každá instance druhé entity může být přiřazena nejvýše k jedné instanci první entity.
- **Vztah N:M** (více k více) **–** zvláštní typ vztahu, kdy libovolné instanci první entity můžeme přiřadit nula, jednu nebo více instancí druhé entity, ale i opačně. V relačním modelu je nutné provést dekompozici tohoto vztahu tak, že definujeme průniková data v průnikové tabulce a relaci typu N:M definujeme jako dvě relace 1:N, přičemž se průniková tabulka nachází v obou těchto relacích na straně N („více“). [6]

2.4.9 Integrita relačního modelu

Kontrola integrity představuje složení několika integritních omezení, která mají zamezit tomu, aby se vytvořená databáze stala nekonzistentní. To znamená, že integrita relačního modelu zajišťuje, aby data uložená v relačním modelu reprezentovala vlastnosti objektů skutečného světa. Rozlišujeme tato integritní omezení: integritní omezení pro entity a integritní omezení pro vztahy entit. [3]

Integritní omezení pro entity jsou následující:

- **Doménová integrita –** znamená omezení pro hodnoty atributů relace. Například datový typ, povinné zadání položky, rozsah hodnot, výchozí hodnota, seznam přípustných hodnot atp. [3]
- **Entitní integrita** - každý entita musí obsahovat právě jeden unikátní nenulový identifikátor, který nazýváme primární klíč. [2]

  Referenční integrita - je obvykle řízena pomocí primárních a cizích klíčů a zaručuje konzistenci a přesnost dat v databázi. [1] *„Pokud existuje v tabulce cizí klíč, musí buď hodnota cizího klíče odpovídat hodnotě některého záznamu v domovské tabulce, nebo musí mít cizí klíč prázdnou hodnotu.“* [2, str. 71]

2.5 Normalizace

Normalizace je proces, jehož výsledkem je množina relací, která splňuje jistou množinu vlastností. S touto technikou přišel v roce 1972 Dr. E. F. Codd. Ten během své práce na teorii relačních databází zjistil, že při aktualizaci dat nenormalizovaných relací se vyskytují určité problémy, které označil výrazem anomálie. Tyto anomálie způsobovaly například nemožnost vložení některých dat, z důvodu umělé závislosti na jiné relaci, nebo nežádoucí ztrátu dat v situaci, kdy odstraněná data jedné entity představovala vlastně i data charakterizující jinou entitu. [6]

Při normalizaci jsou tedy datové struktury upravovány tak, aby byly splněny dané normalizační formy, které vychází z požadavku na efektivní ukládání dat a minimalizují redundance při zachování integrity a konzistence dat. Pokud datový model porušuje některou normalizační formu, tak není navržen optimálně. Při normalizaci na vyšší normalizační úrovni musí být model normalizován na všech úrovních předcházejících. [3]

2.5.1 1. normální forma

„Relace je v první normální formě, pokud jsou všechny její atributy definovány nad skalárními obory hodnot (doménami).“ [3, str. 56]

Zmíněná definice je základní podmínkou této normální formy. Jinými slovy nám říká, že každá buňka tabulky obsahuje právě jednu hodnotu, která nesmí být složená a ani tzv. vícehodnotová. Vyjma možnosti využít složenou hodnotu v konkrétním případě, kdy nepotřebujeme rozlišovat jednotlivé části zadané informace, jako je např. adresa. V tomto případě zůstává 1. normální forma splněna. [6]

2.5.2 2. normální forma

„Relace je ve druhé normální formě, pokud je v první normální formě a navíc všechny její atributy jsou závislé na celém kandidátním (primárním) klíči.“ [3, str. 58]

To znamená, že tato normální forma týká pouze tabulek, které mají složený primární klíč, tzn. primární klíč těchto tabulek je tvořen dvěma a více sloupci. Tabulka v 1NF, jejíž primární klíč tvoří jediný sloupec, je automaticky v 2NF. Definice druhé normální formy ve své podstatě obsahuje požadavek nepokoušet se dvě různé entity vyjádřit v jediné tabulce. Tím zamezíme nežádoucí redundanci a taktéž definujeme mechanismus pro uložení informací, které by nám jinak unikly. [4]

2.5.3 3. normální forma

„Relace je v třetí normální formě, pokud je ve druhé normální formě a navíc všechny její neklíčové atributy jsou vzájemně nezávislé“ [3, str. 60]

Podmínka třetí normální formy spočívá v neexistenci žádné tranzitivní závislosti. To znamená, že všechny neklíčové atributy relace jsou závislé pouze na jejím primárním klíči. Tedy v relaci neexistuje mezi dvěma atributy a klíčem závislost, v které by byl jeden z atributů závislý na druhém atributu a až ten by byl závislý na kandidátním (primárním) klíči. [3]

2.5.4 Boyce-Coddova normální forma

Relace je Boyce-Coddově normální formě, pokud mezi kandidátními klíči neexistuje žádná funkční závislost a to za předpokladu podmínek:

- V relaci musí být dva nebo více kandidátních klíčů
- Minimálně dva z kandidátních klíčů musí být složené
- Kandidátní klíče se v některých atributech musí překrývat

Tato normální forma je považována za variaci třetí normální formy. Platí, že je-li relace v Boyce-Coddově normální formě, je také ve třetí normální formě, ovšem neplatí to naopak. [3]

Pokud si správně uvědomíme, o čem daná relace logicky vypovídá, lze celkem snadno předcházet porušení pravidel Boyce-Coddovy normální formy. Např. jestliže uvažujeme relaci Výrobky, pak v ní nemají co dělat informace o dodavatelích (a naopak). [6]

2.5.5 4. normální forma

„Relace je ve čtvrté normální formě, pokud je v Boyce-Coddově normální formě, a navíc všechny vícehodnotové závislosti jsou zároveň funkčními závislostmi z kandidátních klíčů - v jedné závislosti se nesmí spojovat nezávislé opakované skupiny.“ [3, str. 63]

2.5.6 5. normální forma

K páté normální formě obecně neexistuje žádná standardní definice. Tato norma pracuje se specifickým typem omezení, které nazýváme spojená závislost. Ta vyjadřuje cyklické omezení, které říká, že pokud je entita 1 spojena s entitou 2, entita 2 je spojena

s entitou 3 a entita 3 je spojena zpětně s entitou 1, pak všechny tři entity musí být součástí stejného vektoru hodnot. [6]

2.6 SQL

SQL (Structured Query Language) je dotazovací jazyk, který se používá při komunikaci s relačními databázemi a nejrozšířenější jazyk, který umožňuje vytvářet databázové dotazy. Dotaz je český ekvivalent pro anglické slovo query a znamená požadavek, který se odesílá databázi. Na základě tohoto požadavku poskytne databáze uživateli odpověď. Jazyk SQL se řadí mezi neprocedurální jazyky, u kterých se nemusí definovat, jak požadované výsledky z dat získat. K nejzajímavějším vlastnostem tohoto jazyka patří především jeho deklarativnost - lze ho vložit do jiného programovacího jazyka, který je již procedurální. Existují sice některá procedurální rozšíření jako je PL/SQL společnosti Oracle nebo Transact-SQL v databázích Microsoft. Tato rozšíření ovšem lze označit za nové jazyky. [7]

2.6.1 Historie jazyka SQL

Jazyk SQL byl vyvinut na konci sedmdesátých let dvacátého století v laboratoři společnosti IBM v San Jose v Kalifornii. „Sequel“, jak bývá často jazyk SQL označován, byl prvotně vyvinut pro produkt společnosti IBM s názvem DB2 (což je relační databázový systém, který se používá i v současné době). SQL je prvním neprocedurálním jazykem, do doby jeho vytvoření existovaly pouze procedurální jazyky nebo jazyky třetí generace (3GL), jako je COBOL nebo C. [1]

2.6.2 Data Definition Language (DDL)

Příkazy standardní množiny jazyka SQL se dělí do několika podmnožin. První z nich je jazyk DDL. Pomocí těchto příkazů je možné definovat, vytvářet, měnit a odstraňovat objekty a struktury v relačních databázích. Neumožňují ovšem vkládat ani aktualizovat data, která jsou v objektech uložena. Tři základní klíčová slova DDL:

- CREATE – vytvoří nový databázový objekt uvedeného typu, např. CREATE DATABASE, CREATE TABLE, CREATE INDEX nebo CREATE VIEW.
- ALTER – změní definici existujícího databázového objektu uvedeného typu, nejčastěji se používá ALTER TABLE.
- DROP – odstraní či zruší existující databázový objekt uvedeného typu [4].

2.6.3 Data Manipulation Language (DML)

Příkazy pro manipulaci s údaji slouží pro vkládání, aktualizaci a mazání údajů:

- INSERT - pro vkládání záznamů do tabulky.
- DELETE - pro vymazání záznamů v tabulce.
- UPDATE - pro aktualizace záznamů [4].

2.6.4 Data Query Language (DQL)

Tento jazyk obsahuje jen jeden příkaz a to SELECT, pomocí něhož je možné načítat data z databáze. Výsledkem každého takového příkazu je tabulka dotazu složená z jednoho nebo více sloupců a žádného nebo více řádků. Termínem „příkaz SELECT“ ovšem hodně uživatelů nesprávně označuje libovolné příkazy jazyka SQL, které obsahují klíčové slovo SELECT [7].

Příkaz SELECT se využívá k vyvolání a zobrazení dat z jedné či více tabulek. Jedná se o nejpoužívanější příkaz SQL, který je také mimořádně silným příkazem. Jeho obecná podoba vypadá následovně:

SELECT název sloupce/sloupců FROM název tabulky/tabulek
WHERE podmínka hledání
GROUB BY název sloupce/sloupců HAVING podmínka hledání
ORDER BY název sloupce/sloupců

Pořadí klauzulí není možné měnit, ale povinné klauzule jsou pouze první dvě, tedy SELECT a FROM, ostatní jsou nepovinné. Výsledkem každého příkazu SELECT je tabulka dotazu složená z jednoho nebo více sloupců a žádného nebo více řádků.

Posloupnost zpracování v příkazu SELECT:

- FROM - udává tabulku/tabulky, které se mají použít.
- WHERE - označení podmínky, které definují vlastnosti vybraných řádků.
- GROUP BY - slouží pro seskupení identifikačních polí.
- HAVING - filtrování námi požadovaných informací podle určité podmínky.
- SELECT – specifikuje požadované sloupce, které chceme zobrazit.
- ORDER BY – určuje řazení výstupu. [2]

2.6.5 Data Control Language (DCL)

Tato skupina obsahuje speciální příkazy pro řízení provozu a údržby databáze.

- GRANT - pro přidělování uživatelských oprávnění k objektům
- ALTER USER - pro změnu definice uživatele.
- DROP USER - pro odstranění uživatele.
- REVOKE - odnímání oprávnění. [4]

2.6.6 Transaction Control Commands (TCC)

Do této skupiny patří příkazy pro řízení transakcí, cože je sada příkazů, která je zpracovávána jako nedělitelná jednotka.

- CREATE TRANSACTION - vytvoření transakce.
- START TRANSACTION - zahájení transakce.
- COMMIT - potvrzení transakce.
- ROLLBACK - zrušení transakce a návrat do původního stavu. [4]

2.7 MySQL

MySQL je relační databázový systém. Databáze umožňují efektivně ukládat, hledat, třídit a získávat data. MySQL server poskytuje možnost přístupu více uživatelů zároveň. Používá SQL (Structured Query Language), což je celosvětově používaný standardní dotazovací jazyk pro databáze. MySQL je veřejnosti přístupná od roku 1996, ale její kořeny sahají až do roku 1979. MySQL je také Open Source produktem.

Mezi konkurenty MySQL můžeme zařadit PostgreSQL, Microsoft SQL Server a Oracle. Mezi silné stránky MySQL patří:

- Vysoká rychlost
- Nízké náklady
- Snadné nastavení
- Přenositelnost (funguje na Microsoft i Unix platformách)
- Zdrojový kód je k dispozici [10]

2.8 Datové typy

Pomocí datových typů určujeme, jaký typ dat se může do příslušného atributu uložit. MySQL rozlišuje tři základní datové typy: číselné typy, řetězcové nebo textové typy a kalendářní a časové typy. [10]

2.8.1 Číselné typy

Datových typů pro čísla máme několik a slouží k ukládání čísel. Můžeme pracovat jak s celými čísly, tak s čísly desetinnými.

- INTEGER (zkráceně INT) - celé číslo
  - má několik variací:
    - TINYINT - použití při velmi malých celých číslech
    - SMALLINT - malá celá čísla
    - MEDIUMINT - středně velká celá čísla
    - BIGINT - velká celá čísla
- FLOAT - číslo s plovoucí desetinnou čárkou
- DOUBLE - číslo s plovoucí desetinnou čárkou dvojnásobné přesnosti
- NUMERICAL a DECIMAL (DEC) – pro ukládání přesných hodnot čísel s plovoucí desetinnou čárkou, obvyklé použití u peněžních hodnot [10]

2.8.2 Řetězcové typy

Řetězcové typy slouží především k ukládání textu, ale poradí si s jakýmikoli daty, což znamená, že do nich můžeme jako znaky ukládat i čísla. Datové typy pro tvorbu řetězců mají svojí pevnou délku, některé dokážou i rozlišovat velikost písmen.

- CHAR - řetězec s pevně danou délkou M.
  - za typem CHAR se většinou zadává délka řetězce, např. CHAR(20)
  - není-li délka zadána, je výchozí hodnota automaticky nastavena na 1
  - maximální délka typu CHAR je 255 znaků
  - i když bude ukládaný řetězec kratší než zadaná délka, hodnota typu CHAR bude mít vždy takovou délku, jakou specifikujete, protože se obsah pole doplní mezerami na délku M
  - tyto mezery jsou však při načítání dat z databáze ignorovány

- v případě, že je řetězec ukládaný do databáze delší než M, tak se zkrátí na délku M a zbytek se „ztratí“.
- ukládání hodnot CHAR zabírá na disku více místa než ukládání hodnot v řetězcích s proměnlivou délkou, za to se ale z tabulek, kde mají všechny sloupce pevnou délku (tzn. typy CHAR, NUMERIC nebo DATE), vybírají řádky rychleji

- VARCHAR - řetězec s proměnlivou délkou M
  - za typem je do závorek nutné zadat specifikaci délky, např. VARCHAR(10)
  - rozsah od 0 do 225 znaků
  - je-li řetězec ukládaný do databáze kratší než M, koncové mezery se při ukládání odstraní a na disku se nezabírá tolik místa jako u typu CHAR
  - v případě, že je ukládaný řetězec delší než M, tak se zkrátí na délku M a zbytek se „ztratí“.
- TEXT – pro ukládání delších částí textu, které se nevejdou do datových typů CHAR nebo VARCHAR
  - má několik variací:
  - TINYTEXT - malá hodnota TEXT
  - TEXT - normální hodnota TEXT
  - MEDIUMTEXT - středně velká hodnota TEXT
  - LONGTEXT - velká hodnota TEXT
- BLOB – zkratka pro Binary Large OBject
  - stejný typ jako TEXT s rozdílem, že BLOB není k ukládání textu, ale binárních dat
  - má také několik variací:
  - TINYBLOB - malá hodnota BLOB
  - MEDIUMBLOB - středně velká hodnota BLOB
  - LONGBLOB - velká hodnota BLOB
- ENUM – výčtový typ pro uchovávání množiny možných hodnot
  - každý řádek může obsahovat jednu hodnotu z výčtové množiny, příklad deklarace: pohlavi enum(‘m’, ‘ž’)
- SET – stejný typ jako ENUM s rozdílem, že řádky mohou obsahovat nula nebo více z výčtové množiny [10]

2.8.3 Typy pro datum a čas

Některé datové typy spojují datum a čas dohromady, v jiných datových typech je datum a čas odděleně. Existují také speciální časové značky pro zobrazování aktuálního času. Samozřejmě lze využít i datový typ, který zaznamenává pouze roky.

- DATE – pro ukládání kalendářního data
  - ve formátu rok-měsíc-den ‘YYYY-MM-DD‘
- TIME - čas ve formátu hodiny-minuty-sekundy ‘HH:MM:SS’
- DATETIME – kombinace předchozích dvou typů
  - datum a čas ve formátu ‘CCYY-MM-DD HH:MM:SS’
- TIMESTAMP - časová značka ve formátu ‘CCYY-MM-DD HH:MM:SS’
  - pokud je upraven jakýkoliv sloupec v řádku, kde je nastaven TIMESTAMP, upraví se i sloupec a to na aktuální datum a čas, kdy byla změna provedena
  - to platí i v případě, nastavíte-li sloupec, který je typu TIMESTAMP, na hodnotu NULL, tak bude taktéž uchovávat časový údaj okamžiku, kdy byl řádek naposledy vložen či změněn
- YEAR - hodnoty udávající rok
  - Např. YEAR(2) nebo YEAR(4) dle toho, zda požadujete dvou nebo čtyřmístný formát roku
  - YEAR(2) představuje pouze roky v rozsahu od 1970 do 2069
  - standardně se používá YEAR(4) – je to výchozí hodnota [10]

3 ANALÝZA SOUČASNÉHO STAVU

3.1 O společnosti DATEL PLUS s.r.o.

Společnost DATEL PLUS s.r.o. byla založena roku 1997 dvěma společníky a to manželi Jitkou a Ludvíkem Dvořákovými. Nabídka služeb společnosti se během několika let rozrostla o zřizování, montáž, údržbu a servis telekomunikačních zařízení, poskytování technických služeb k ochraně majetku a osob, vedení účetnictví a zprostředkovatelskou činnost.

V současnosti společnost poskytuje komplexní služby obohacené o mnohaleté zkušenosti v oblasti bezpečnostní, informační a komunikační technologie a také v oblasti daňového poradenství, účetnictví a personalistiky.

3.2 Organizační struktura

V současné době má společnost 7 kmenových zaměstnanců. Některé služby jsou řešeny s výpomocí spolupracujících firem (např. zabezpečovací systémy).

Firma je rozdělena na dvě sekce:

- Účetní a daňová kancelář
- Technické oddělení

3.2.1 Účetní a daňová kancelář

První ze dvou sekcí se zabývá vedením účetnictví pro firmy a živnostníky, personalistikou a daňovým poradenstvím.

3.2.2 Technické oddělení

Náplní práce této sekce je instalace, montáž a servis komunikačních a bezpečnostních systémů, návrh, dodávka a správa IT vybavení firem či správa sítí.

3.3 Současné vedení skladové evidence

Společnost používá ekonomický systém Pohoda ve verzi Komplet od firmy Stormware, v kterém vede i skladovou evidenci. Mimo systém Pohoda společnost vzhledem k jejímu zaměření v oblasti IT ukládá data o zboží, a to především o jeho pohybu v rámci společnosti, taktéž ve vlastní databázi vytvořené v Microsoft Accessu.

3.3.1 Ekonomický systém Pohoda

Ekonomický a účetní systém Pohoda vyvinula společnost Stormware, která působí na českém trhu od roku 1996. Pohoda je jen jedním z několika softwarů, jež firma za období své existence vytvořila, ale patří mezi ty nejúspěšnější a nejžádanější. Program je možné zakoupit ve třech aktuálních řadách roku 2015, z nichž každá řada obsahuje několik variant výběru. Tyto řady i jednotlivé varianty se liší v ceně a v množství nabízených modulů a požadovaných agend. Výhodou je tedy možnost jednotlivých podniků zakoupit produkt podporující ty oblasti, které bude v budoucnu ve svém účetnictví využívat. [12]

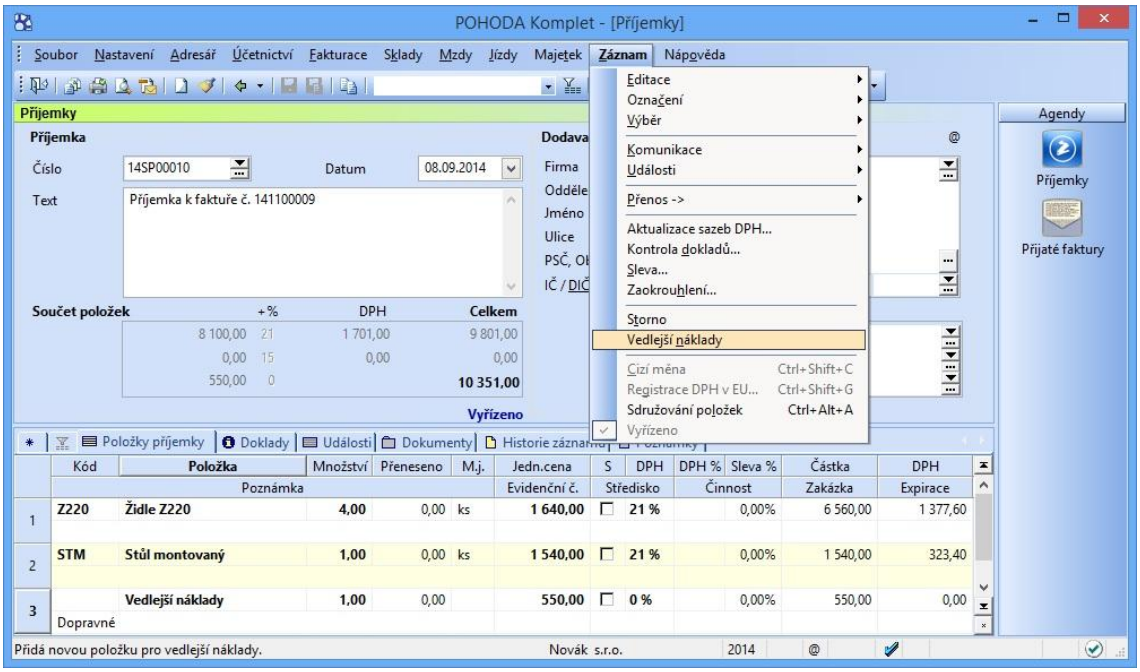
3.3.2 Analýza systému

Tato kapitola se bude zabývat popisem jednotlivých agend, které účetní program Pohoda nabízí. Po účel analýzy systému jsem využil také instalaci zkušební bezplatné verze Pohoda Start 2012.

Hlavní moduly:

- *Fakturace* – vytváření a zpracovávání odběratelských, dodavatelských a zálohových faktur
- *Sklady* – řízení zásob
- *Personalistika* – zaznamenávání a uchovávání údajů o zaměstnancích, jejich platových výměrech, srážkách, spoření, slevách na dani atd.
- *Majetek* – evidence majetku, výpočet a zaúčtování daňových či účetních odpisů

Skladová evidence systému Pohoda funguje jako prostředek pro řízení zásob. Skladové agendy poslouží pro reálné vedení skladů či pro snadné vystavování položkových dokladů. To poskytuje podrobný přehled o skladových položkách a finančních prostředcích, uložených v zásobách. [13]



Obrázek č. 2: Pohoda – Příjemky (24)

Pohoda umožňuje vedení neomezeného počtu skladů a členění dle vlastních potřeb. Podporuje:

- evidenci zboží, materiálu, služeb
- záruky, šarže a výrobní čísla
- čárové kódy pro vkládání skladových položek do jednotlivých dokladů a pro sestavení inventury
- slevy, cenové hladiny, individuální ceníky
- účtování zásob způsobem A nebo B
- zpracování inventur
- sledování reklamací a oprav
- prodej zboží přes internet
- automatické objednávání zásob
- používání náhledů zboží

Každé skladové položce je možné přiřadit vlastní kód, zadat údaje o dodavateli a výrobci, měrných jednotkách, hmotnosti a objemu nebo také nastavit limit pro minimální a maximální množství na skladě a DPH při nákupu a při prodeji. [13]

3.3.3 Zhodnocení

Současný systém vedení skladové evidence v programu Pohoda slouží k přehlednému zaznamenávání informací o příjemkách, výdejkách a skladových kartách. Tento systém ovšem není tak podrobný, není možné v něm evidovat podrobnou historii pohybu zboží a je nutné vést zvláštní evidenci v Microsoft Accessu, což je časově neefektivní. S ohledem na nedostačující evidenci tohoto systému vznikla ve firmě potřeba použití jiného systému pro správu skladové agendy, která by byla flexibilnější a splňovala nároky společnosti.

3.4 Účel databáze

Hlavním účelem návrhu databáze je především nahrazení dvojího vedení skladové evidence a tím pádem zefektivnění práce se všemi evidovanými daty.

3.5 Požadavky na databázi

Výstupy provedených rozhovorů s IT specialistou umožňují definovat hranice systému a uživatele databáze. Databázový systém společnosti bude pokrývat jak oblast nákupu zboží, tak i následný prodej cílovému zákazníkovi.

Do databáze budou tedy ukládána data o nákupu a prodeji, tj. záznamy o zboží, jeho příjmu na sklad, výdeje ze skladu, pohybu zboží v rámci společnosti, jednotlivých dodavatelích a odběratelích, či zaměstnancích. Výsledný systém by neměl zavádět zbytečné funkce, které se nevyužijí a které by byly pro uživatele spíše nepřehledné než užitečné.

3.6 Používané databázové platformy

Relační databáze mají na trhu několik desítek databázových systémů. Za účelem vhodného výběru pro realizaci požadované databáze budou v této kapitole představeny nejznámější a nejvyužívanější z nich.

3.6.1 Oracle Database

Oracle Database je databázový systém dostupný pro všechny platformy také s podporou 64 bitové architekury (vyjma neplacené verze Express Edition). Používá standardizovaný dotazovací jazyk SQL-92, který rozšiřuje o vlastní funkčnosti

především pro správu dat, imperativní programovací jazyk PL/SQL a další možnosti pro snazší správu a řízení databáze. [11]

Databáze Oracle je nabízena ve čtyřech verzích (edicích):

- *Express Edition* – bezplatná základní verze, omezená na 1 jádro CPU, 1GB RAM a 4 GB dat, zahrnuje pouze některá možná Oracle rozšíření pro správu databází, na serveru je možné současně provozovat pouze jednu instanci databáze
- *Standard Edition One* – již licencovaná verze, cena se odvíjí od použitých procesorů, omezuje pouze maximální počet jader CPU a to dvě, zahrnuje více rozšiřujících Oracle funkčností
- *Standard Edition* – podpora až čtyř CPU jader a zahrnuje více funkčností pro podporu škálovatelnosti
- *Enterprise Edition* – neomezena počtem procesorů ani podporou rozšiřujících Oracle funkčností, které jsou již vestavěné či je možnost je volitelně dokoupit

Společnost Oracle je jednou z nejvýznamnějších společností působící na trhu v oblasti databázových řešení. Systém je používán především ve státní správě a ve velkých firmách, kde je potřeba zajistit co nejrobustnější řešení. Díky tomuto využití je zde patrný neustálý vývoj. Nejnovější verzí databázového systému je verze 12c. [14]

3.6.2 MS SQL Server

Microsoft SQL Server je relační databázový systém považovaný za bezpečný a snadno škálovatelný. Možnost provozování je ovšem omezena na operační systémy MS Windows. Aktuální verze tohoto systému je MS SQL Server 2014. Od verze 2005 je přímo vestavěná podpora .NET, čímž Microsoft nabízí komplexní služby pro vývojáře. Tento databázový systém rovněž nabízí rozšíření jazyka SQL – tzv. Transact-SQL, který umožňuje definovat uživatelské funkce pomocí procedurálního programování. [15]

Je nabízený ve třech hlavních (komerčních) edicích:

- *Standard Edition* - základní databázové funkčnosti včetně reportování a analytických nástrojů, omezena na 16 procesorových jader

- *Business Intelligence* - zaměřena na rychlé vyhledávání dat (funkce Power View umožňuje vizuální procházení dat)
- *Enterprise Edition* - doplňuje nové nástroje pro řízení bezpečnosti a vysoké dostupnosti (např. funkce ColumnStore) [16]

Dále jsou k dispozici „nízkorozpočtové“ edice:

- *Developer Edition* - verze Enterprise určená pouze pro účely vývoje a testování
- *Web Edition* – levnější edice dostupná pouze poskytovatelům služeb s podepsanou smlouvou SPLA (Service Provider License Agreement), tj. hostery, outsourcery
- *Express* - verze zdarma s technickými omezeními velikosti databáze, velikosti využitelné operační paměti RAM a podporovaných CPU

3.6.3 MySQL

MySQL je víceplatformový relační open source databázový systém. Je dostupný pro operační systém Linux, systémy založené na Unixu, OS/2 i Microsoft Windows. Tento systém vytvořila firma MySQL AB a v současnosti je vyvíjen a vlastně firmou Oracle Corporation. Je nabízen jako součást softwarového balíčku LAMP (Linux, Apache, MySQL, PHP), který se využívá pro tvorbu webových aplikací. [17]

Dříve byla MySQL vytýkána absence transakcí, triggerů, cizích klíčů nebo uložených procedur. Od verze 5.0 však MySQL tyto principy podporuje. Aktuální verzí databázového systému je verze 5.6.

MySQL je nabízen v několika edicích.

- *Community Edition* – základní edice, která je volně dostupná
- *Standard Edition*
- *Enterprise Edition*
- *Cluster CGE*

Poslední tři zmíněné edice jsou komerční a placené verze, které se liší různými rozšiřujícími funkcemi a nástroji. [18]

3.7 Zhodnocení a výběr vhodné platformy

Z výše uvedených databázových systémů byl i po poradě se zadavatelem vybrán databázový systém MySQL. Prvním důvodem pro tuto volbu bylo, že server společnosti běží na operačním systému Linux, tudíž Microsoft SQL Server nepřicházel v úvahu. Při dalším rozhodování mezi Oracle Database a MySQL byly brány v úvahu požadavky na hardware, rychlost systému a taktéž náklady na pořízení.

Oracle Database je velmi robustní databázový systém užívaný i na nejzatíženějších systémech. Oproti MySQL má ale poměrně vysoké HW požadavky a taktéž velmi vysoké náklady na verze systému pro náročnější aplikace. Je zde sice možné použití bezplatné verze Express, ta je ale značně omezená (umožňuje využívat pouze 1 jádro procesoru, 1GB RAM a velikost datových souborů nemůže překročit velikost 4 GB).

Oproti tomu u MySQL není používání v základní verzi nijak limitováno, databázový server je možné spustit prakticky na všech osobních počítačích, které jsou v současnosti dostupné. Mezi hlavní výhody tohoto systému patří jeho multiplatformost, také vysoký výkon, rychlost a relativní jednoduchost. Další výhodou je taktéž skutečnost, že je jednou z klíčových součástí tzv. LAMP (Linux, Apache, MySQL, PHP), která se v současné době používá u velkého počtu webových aplikací, protože celé toto řešení je velice spolehlivé. Jeden z hlavních důvodů tohoto výběru byl i fakt, že společnost má v plánu na základě navrhnuté databáze vytvořit webovou aplikaci a postavení na open source základu výrazně snižuje finanční náklady na vývoj i běh aplikace.

4 VLASTNÍ NÁVRHY ŘEŠENÍ

V této části vytvořím návrh řešení k daným požadavkům na databázi pro zadavatelskou společnost. Nejprve bude vytvořen konceptuální návrh, v kterém budou identifikovány všechny důležité objekty, které bude v databázi nezbytné reprezentovat, a relace mezi nimi. To povede ke vzniku ER diagramu. Následně vytvořím logický návrh databáze, v kterém budou znázorněny všechny tabulky a jejich klíče. Třetím krokem bude fyzický návrh databáze, v kterém jde o samotné vytvoření tabulek, procedur a spouští v jazyce SQL.

4.1 Konceptuální návrh

Nejprve bude třeba identifikovat všechny entity a relace v databázi, aby mohl být vytvořen ER diagram, který bude splňovat požadavky na databázi.

4.1.1 Identifikace entit

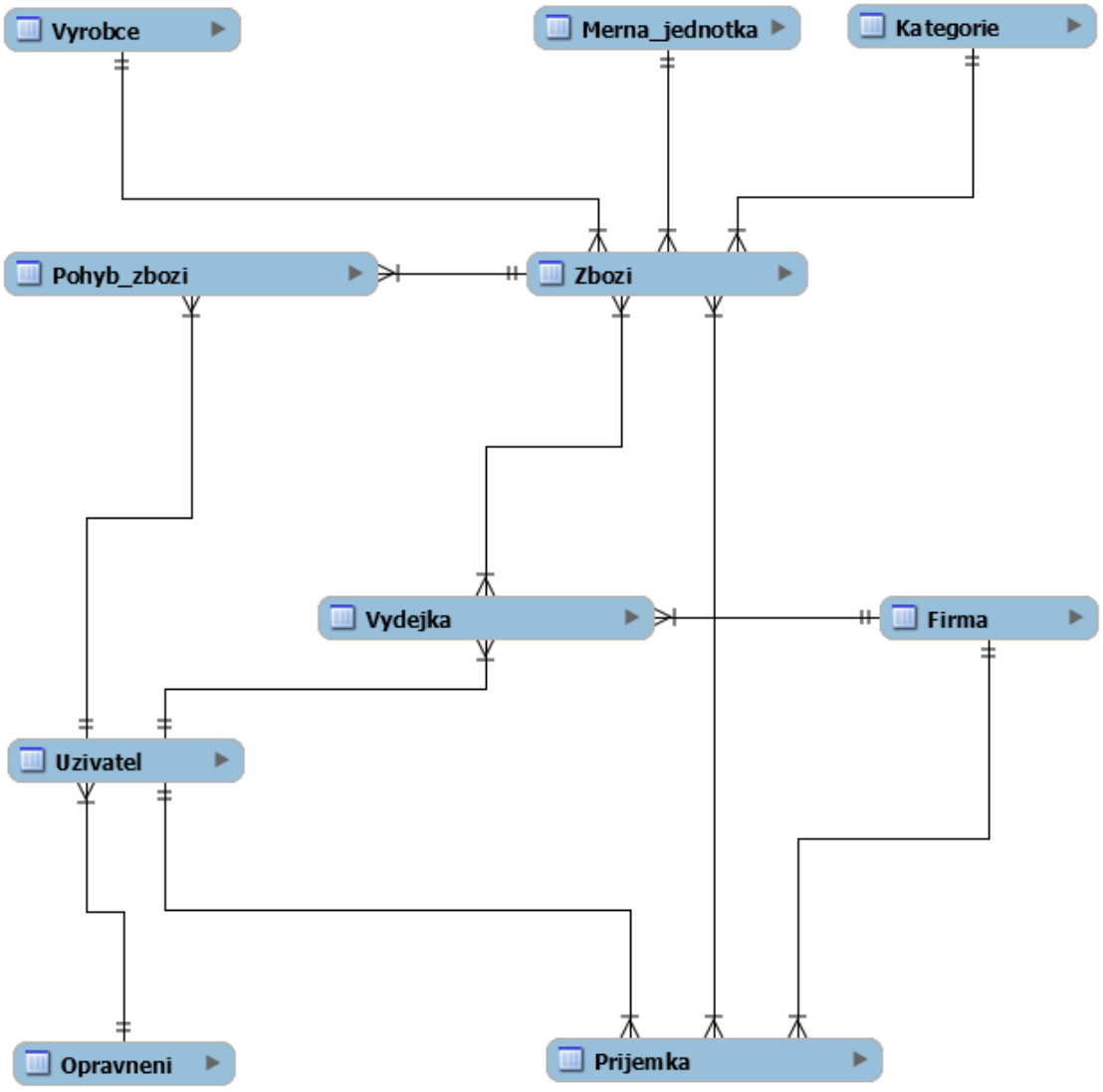
Určení nejdůležitějších objektů, se kterými uživatelé pracují, je první fáze při vytváření konceptuálního datového modelu. Tyto objekty jsou entitními typy pro tento model.

Název entity	Popis
Zboží	Zboží, s kterým společnost obchoduje
Příjemka	Záznam o převzetí zboží do skladu
Výdejka	Záznam o vydání zboží ze skladu pro prodej
Výrobce	Výrobce konkrétního zboží
Firma	Firma, která se společností obchoduje, ať už jako odběratel nebo dodavatel
Kategorie	Kategorie jednotlivého zboží
Měrná jednotka	Měrná jednotka, v které je konkrétní zboží ve skladu vedeno
Uživatel	Konkrétní zaměstnanec ve společnosti
Pohyb zboží	Informace o pohybu zboží ve společnosti před finálním prodejem
Oprávnění	Oprávnění uživatelů

Tabulka č. 1: Identifikace entit (Zdroj: Vlastní zpracování)

K modelování databáze bude využit program MySQL Workbench 6.2 CE, což je open source nástroj od autorů MySQL pod licencí GPL, který slouží nejen k vizuálnímu návrhu databáze. Umožňuje vytvářet schémata tabulek spolu s jejich sloupci a atributy a také relace mezi jednotlivými tabulkami.

Po identifikaci konkrétních objektů neboli entit vznikl následující model, který zachycuje statický pohled na danou situaci. Vztahy vyjadřují fyzickou nebo konceptuální vazbu mezi entitami.



Obrázek č. 3: ER diagram - pouze entity (Zdroj: vlastní zpracování)

4.1.2 Identifikace relací

Entity	Relace	Popis vztahu
Výrobce – Zboží	1:N	Kolik vyrábí výrobce zboží? N. Jedno zboží je vyráběno kolika výrobci? 1.
Kategorie – Zboží	1:N	Kolik zboží zahrnuje jedna kategorie? N. Jedno zboží patří do kolika kategorií? 1.
Měrná jednotka - Zboží	1:N	Kolik zboží se vyskytuje u měrné jednotky? N. Jedno zboží má kolik měrných jednotek? 1.
Zboží – Příjemka	M:N	Kolik zboží obsahuje jedna příjemka? N. Jedno zboží se nachází v kolika příjemkách? N.
Uživatel – Příjemka	1:N	Kolik příjemek provádí uživatel? N. Jedna příjemka je prováděna kolika uživateli? 1.
Firma – Příjemka	1:N	Kolik příjemek obsahuje dodavatele? N. Jedna příjemka je na kolik dodavatelů? 1.
Zboží – Výdejka	M:N	Kolik zboží obsahuje jedna výdejka? N. Jedno zboží se nachází v kolika výdejkách? N.
Uživatel – Výdejka	1:N	Kolik výdejek provádí uživatel? N. Jedna výdejka je prováděna kolika uživateli? 1.
Firma – Výdejka	1:N	Kolik výdejek obsahuje dodavatele? N. Jedna výdejka je na kolik dodavatelů? 1.
Zboží - Pohyb zboží	1:N	Kolik pohybů má jedno zboží? N. Jeden pohyb se vyskytuje u kolika zboží? 1.
Uživatel - Pohyb zboží	1:N	Kolik pohybů zboží provádí jeden uživatel? N. Jeden pohyb je prováděn kolika uživateli? 1.
Oprávnění - Uživatel	1:N	Kolik uživatelů má přiděleno jedno oprávnění? N. Jeden uživatel má přiděleno kolik oprávnění? 1.

Tabulka č. 2: Identifikace relací (Zdroj: Vlastní zpracování)

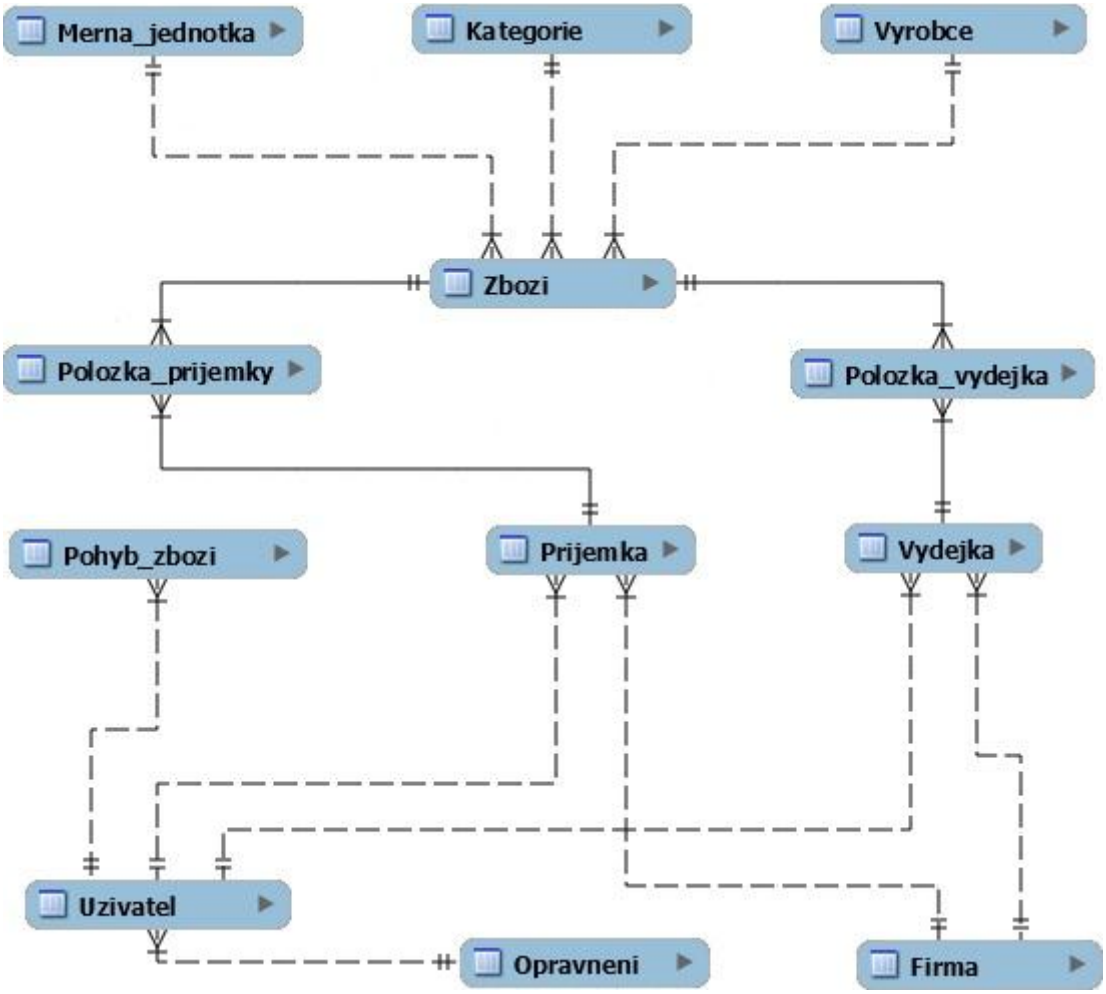
V tabulce č. 2 se nachází dvě vazby M:N. Jde o entity *Zboží* - *Příjemka* a *Zboží* - *Výdejka*. Proto je nutno provést dekompozici vztahu. V případě entit *Zboží* - *Příjemka* vytvořím novou tabulku *Položka příjemky* a vložím do ní primární klíče obou entit. Tím vztah M:N dekomponuji na 1:N a N:1. Stejným způsobem dekomponuji vztah mezi entitami *Zboží* - *Výdejka*, kde bude vytvořena nová tabulka *Položka výdejky*.

Obrázek č. 4: Dekompozice vztahu M:N – zboží:příjemka (Zdroj: vlastní zpracování)

Obrázek č. 5: Dekompozice vztahu M:N – zboží:výdejka (Zdroj: vlastní zpracování)

4.1.3 Základní ER diagram

Po dekompozici M:N vztahů vznikl konceptuální model, který představuje základní ER diagram. Ten bude v další fázi převeden na logický model, kdy budou vytvořeny normalizované tabulky.



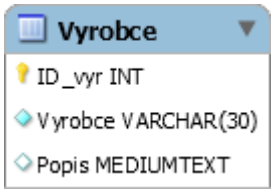
Obrázek č. 6: ER diagram – po dekompozice vztahů M:N (Zdroj: vlastní zpracování)

4.2 Logický návrh

V rámci logického návrhu databáze se z konceptuálního modelu vytvoří tabulky, definují datové typy jednotlivých atributů a jejich integritní omezení. Během této fáze bude provedena kontrola normalizace tabulek v databázi.

4.2.1 Výrobce

Do tabulky *Výrobce* se ukládají informace o jednotlivých výrobcích. Není potřeba nic více než název a případně popis výrobce. ID výrobce je primárním klíčem, který je vždy NOT NULL (nesmí mít prázdnou hodnotu).



Obrázek č. 7: Tabulka Výrobce (Zdroj: vlastní zpracování)

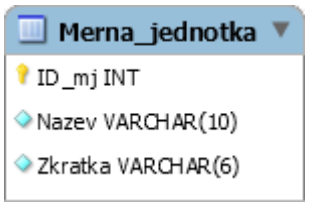
4.2.2 Kategorie

Touto tabulkou je řešeno členění zboží do kategorií. Primárním klíčem je ID kategorie, dalšími atributy jsou kategorie s omezením NOT NULL a nepovinným atributem popis (může obsahovat prázdné hodnoty).

Obrázek č. 8: Tabulka Kategorie (Zdroj: vlastní zpracování)

4.2.3 Měrná jednotka

V této tabulce jsou rozlišovány měrné jednotky, v které je zboží ve skladu vedeno. Primárním klíčem je zde ID_mj. Dále je evidován název a zkratka, které nesmí mít prázdnou hodnotu (omezení NOT NULL).



Obrázek č. 9: Tabulka Měrná jednotka (Zdroj: vlastní zpracování)

4.2.4 Zboží

Primárním klíčem této tabulky je ID zboží, které zajišťuje jeho jednoznačnou identifikaci. U zboží je potřeba evidovat název zboží, jeho produktové číslo, záruční dobu či popis konkrétního zboží. Dále je tabulka provázána s tabulkami *Výrobce*, *Kategorie* a *Měrná jednotka*. Prostřednictvím jejich primárních klíčů (ID_vyr, ID_kat, ID_mj), které jsou zde použity jako cizí klíče.

Obrázek č. 10: Tabulka Zboží (Zdroj: vlastní zpracování)

4.2.5 Oprávnění

V této tabulce jsou uložena jednotlivá oprávnění, která se následně přidělují uživatelům. Primárním klíčem je zde ID oprávnění. Dalšími atributy jsou druh s omezením NOT NULL a nepovinný atributem popis.

Obrázek č. 11: Tabulka Oprávnění (Zdroj: vlastní zpracování)

4.2.6 Uživatel

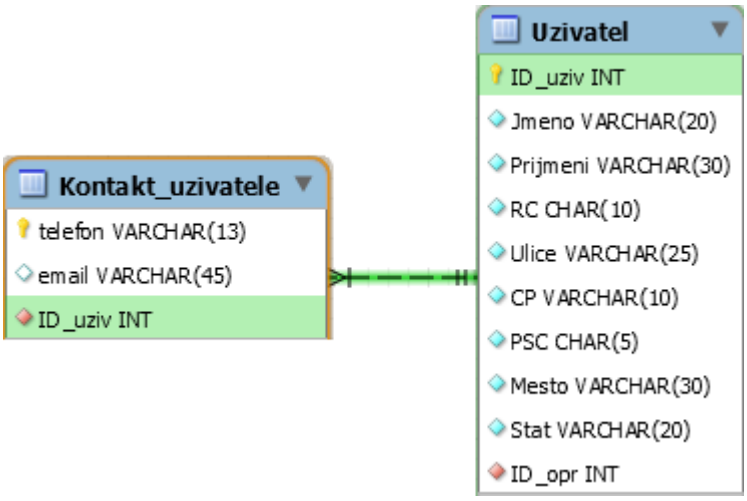
O každém uživateli je potřeba uchovávat informace jako je jeho jméno, rodné číslo, bydliště, telefon či email.

Z hlediska normalizace zde není splněna 1. normální forma, jelikož atribut *jméno* bude obsahovat složené hodnoty - jméno a příjmení (*„Jan Novák“*). A pokud bychom chtěli například vypsat všechny uživatele seřazené abecedně dle příjmení, bylo by to poněkud složitější. Z toho důvodu je správné vytvořit dva samostatné atributy *jméno* a *příjmení*.

V případě, že bychom chtěli vyhledat například pouze uživatele, kteří bydlí v Brně, tak tabulka pořád nesplňuje 1. normální formu, protože atribut *bydliště* taktéž obsahuje složené hodnoty – v tomto případě ulici, č. p., PSČ a město. Vytvořím tedy jednotlivé atributy *ulice*, *č. p.*, *PSČ* a *město*.

Každý uživatel může mít více telefonů (např. služební, osobní) a také více emailů. Z tohoto pohledu tabulka opět nesplňuje 1. normální formu, protože atribut telefon, resp. email by mohl obsahovat více telefonních čísel, resp. emailových adres, tzn. neatomické hodnoty. Aby byla tabulka v 1NF, bude vytvořena samostatná tabulka *Kontakt uživatele*, která bude s tabulkou uživatele propojena pomocí ID uživatele, které je primárním klíčem tabulky *Uživatel*. Mezi těmito tabulkami vznikne vazba 1:N - jeden uživatel může mít více kontaktů. Primárním klíčem tabulky *Kontakt uživatele* je atribut telefon.

Přes *ID oprávnění* je každému uživateli přiděleno oprávnění prostřednictvím provázání na tabulku *Oprávnění*.



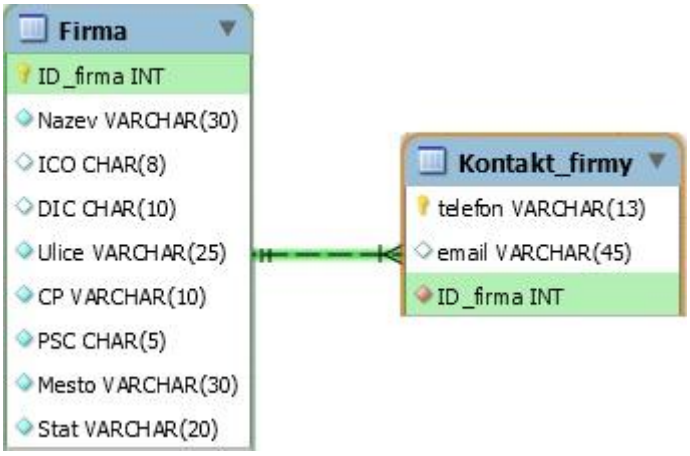
Obrázek č. 12: Tabulka Uživatel a Kontakt uživatele (Zdroj: vlastní zpracování)

4.2.7 Firma

Jak již bylo popsáno v konceptuálním návrhu, tato tabulka představuje jednotlivé odběratele či dodavatele společnosti. Z důvodu toho, že odběratel může být zároveň i dodavatelem společnosti, je tabulka *Firma* společná pro obě možnosti, aby se předešlo redundantním záznamům. U těchto subjektů je nutné ukládat jejich název, sídlo, IČO, DIČ, telefonní čísla a emailové adresy.

Co se týče normalizace, tak je tu obdobná situace jako u tabulky U*živatel*. Aby tabulka splňovala 1. normální formu, atribut *sídlo* rozdělím do více atributů: *ulice, č. p., PSČ, město* a *stát*.

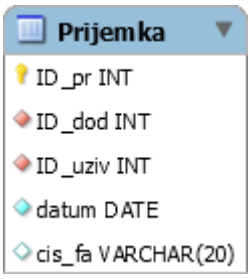
Na každou firmu může být více telefonních čísel či emailových adres. Z toho důvodu bude opět vytvořena nová tabulka, v tomto případě *Kontakt firmy,* která bude s tabulkou firmy propojena pomocí ID firmy, které je primárním klíčem tabulky *Firma*. Vzniklá vazba je 1:N – jedna firma může mít více kontaktů.



Obrázek č. 13: Tabulka Firma a Kontakt firmy (Zdroj: vlastní zpracování)

4.2.8 Příjemka

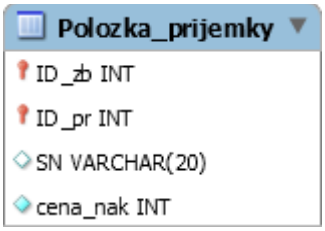
Jednoznačným identifikátorem je zde primární klíč ID příjemky. U každé příjemky je nutné evidovat, kdo je dodavatelem, kdo příjemku vystavil, datum vystavení a číslo faktury. Tabulka je provázána s tabulkami *Firma* a *Uživatel* prostřednictvím jejich primárních klíčů ID_firma a ID_uziv, které jsou zde použity jako cizí klíče ID_dod a ID_uziv.



Obrázek č. 14: Tabulka Příjemka (Zdroj: vlastní zpracování)

4.2.9 Položka příjemky

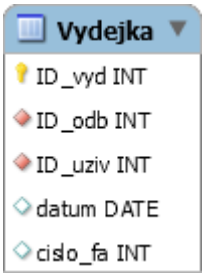
U této tabulky je důležité ukládat, o jaké zboží se jedná a které příjemky je položka součástí. To je zajištěno provázáním s tabulkami *Zboží* a *Příjemka* prostřednictvím jejich primárních klíčů ID_zb a ID_pr, které jsou zde použity jako cizí klíče a společně tvoří složený primární klíč tabulky *Položka příjemky*. Dalšími atributy jsou sériové číslo a nákupní cena položky.



Obrázek č. 15: Tabulka Položka příjemky (Zdroj: vlastní zpracování)

4.2.10 Výdejka

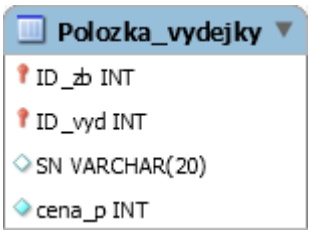
Primárním klíčem této tabulky je ID výdejky, které zajišťuje jeho jednoznačnou identifikaci. U každé výdejky je potřeba evidovat, kdo je odběratelem, kdo výdejku vystavil, datum vystavení a číslo faktury. Tabulka je provázána s tabulkami *Firma* a *Uživatel* prostřednictvím jejich primárních klíčů ID_firma a ID_uziv, které jsou zde použity jako cizí klíče ID_dod a ID_uziv.



Obrázek č. 16: Tabulka Výdejka (Zdroj: vlastní zpracování)

4.2.11 Položka výdejky

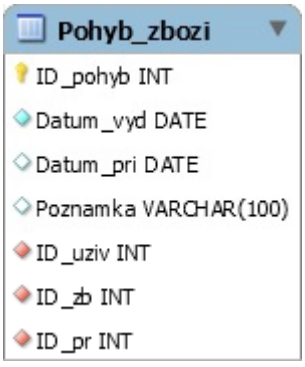
U této tabulky je důležité ukládat, o jaké zboží se jedná a které výdejky je položka součástí. To je zajištěno provázáním s tabulkami *Zboží* a *Výdejka* prostřednictvím jejich primárních klíčů ID_zb a ID_pr, které jsou zde použity jako cizí klíče a společně tvoří složený primární klíč tabulky *Položka výdejky*. Dalšími atributy jsou sériové číslo a prodejní cena položky.



Obrázek č. 17: Tabulka Výdejka (Zdroj: vlastní zpracování)

4.2.12 Pohyb zboží

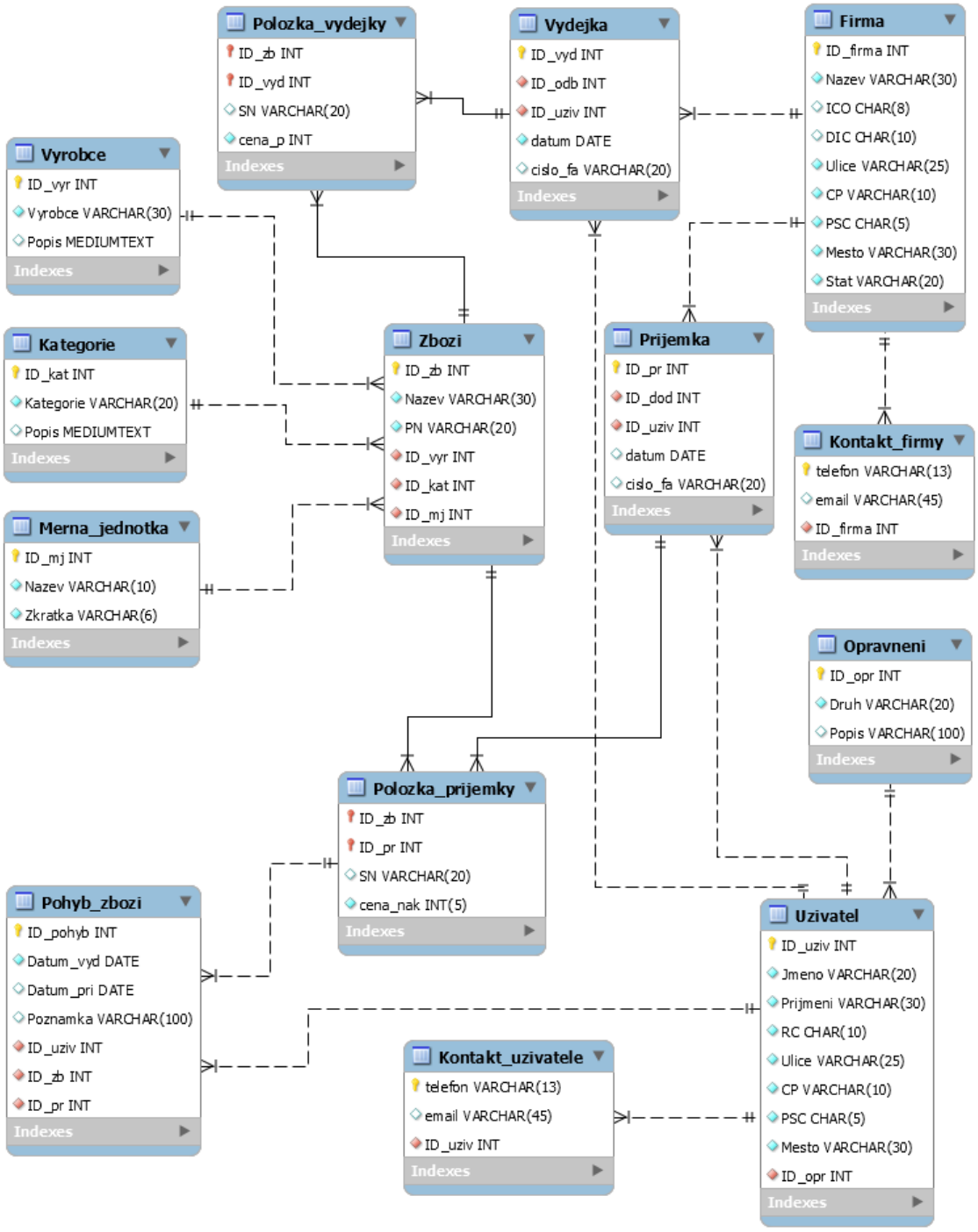
V této tabulce jsou uloženy jednotlivé pohyby zboží v rámci společnosti, které provádí zaměstnanci, resp. uživatelé. Primárním klíčem je zde ID pohybu. Dalšími atributy jsou datum vydání, datum příjmu (vrácení) zpět na sklad a poznámka. Taktéž je potřeba uložit, který zaměstnanec daný pohyb provedl. To je zajištěno provázáním s tabulkou *Uživatel* prostřednictvím jejího primárního klíče *ID_uziv,* který je zde použit jako cizí klíč. Dále je nutné zajistit provázání na tabulku *Položka příjemky,* aby bylo jasné, o jaké zboží se přesně jedná, pomocí použití jejího složeného primárního klíče *ID_zb, ID_pr*.



Obrázek č. 18: Tabulka Výdejka (Zdroj: vlastní zpracování)

4.2.13 Finální ER diagram

Po vytvoření normalizovaných tabulek vznikla datová struktura, kterou představuje finální ER diagram, jež bude implementován v cílovém DBMS.



Obrázek č. 19: ER diagram - logické schéma (Zdroj: vlastní zpracování)

4.3 Fyzický návrh

V této části bude realizována samotná databáze v databázovém systému MySQL. Pomocí jazyka SQL budou vytvořeny tabulky databáze, následně pohledy a procedury. Sada SQL příkazů bude spuštěna v předem připravené prázdné databázi v MySQL Workbench, kde vznikne testovací struktura databáze.

4.3.1 Vytvoření tabulek

Zdrojový kód pro tvorbu tabulek je vzhledem k jeho obsáhlosti uveden v příloze. Jako příklad uvedu vytvoření tabulky zboží. Výklad příkazu CREATE TABLE začíná jednoduchou definicí tabulky *zbozi*. Dále je definován seznam sloupců, odpovídajících typů dat a omezení typu NOT NULL (vložení prázdné hodnoty). Použité datové typy *int*, resp. *varchar(n)* označují celá čísla, resp. řetězec znaků délky nejvýše *n*. AUTO_INCREMENT u sloupce *id_zb* znamená automatické vložení hodnoty o jednu větší než byla hodnota posledně vloženého záznamu. Dále je definován primární klíč tabulky a taktéž cizí klíče. Před samotným vytvořením této tabulky musí být již vytvořeny tabulky *Vyrobce*, *Kategorie* a *Merna_jednotka*, na které odkazují cizí klíče (*id_vyr*, *id_kat*, *id_mj*).

```
CREATE TABLE zbozi (
    id_zb int NOT NULL AUTO_INCREMENT,
    nazev varchar(30) NOT NULL,
    pn varchar(20) NOT NULL,
    zarucni_doba int NOT NULL,
    id_vyr int NOT NULL,
    id_kat int NOT NULL,
    id_mj int NOT NULL,
PRIMARY KEY (id_zb),
FOREIGN KEY (id_vyr) REFERENCES vyrobce (id_vyr),
FOREIGN KEY (id_kat) REFERENCES kategorie (id_kat),
FOREIGN KEY (id_mj) REFERENCES merna_jednotka (id_mj));
```

4.3.2 Vložení testovacích dat

Pro potřeby testování databáze byly jednotlivé tabulky byly naplněny daty. Zde uvedu ukázku vložení dat do tabulky zboží. Výklad příkazu INSERT INTO začíná opět jednoduchou definicí tabulky *zbozi*. Dále jsou definovány seznamy sloupců (atributů), do nichž chceme data vložit. V tomto případě jde o všechny atributy tabulky *zbozi* kromě atributu *id_zb*, který je primárním klíčem této tabulky a má nastaveno automatické vkládání hodnoty při vložení každého záznamu.

```
INSERT INTO zbozi (id_vyr, id_kat, id_mj, nazev, popis, pn, zar_doba) VALUES
(1, 1, 1, 'WD Caviar Blue EX - 1TB', 'Rychlý pevný disk s vysokou kapacitou 1 TB, 64 MB vyrovnávací paměť, 7200 ot/min, rozhraní SATA 6Gb/s.', 'WD10EZEX', 2),
```

4.3.3 Pohledy

Pohledy se využívají pro vytvoření podmnožiny dat z jedné či více tabulek, což umožňuje uživatelům databáze snazší přístup k požadovaným údajům bez použití komplikovaných složených příkazů jazyka SQL.

První pohled slouží k přehledu skladovaného zboží. Zobrazuje název zboží, jeho produktové číslo, kategorii, do které zboží patří, výrobce a měrnou jednotku. Druhý pohled zobrazí všechny společnosti, s kterými je obchodováno, ať už jako s odběratelem nebo dodavatelem, spolu s jejich kontakty. Třetí pohled slouží k výpisu zaměstnanců společnosti a jejich kontaktů. Další pohled zobrazí všechny dodavatele společnosti.

```
CREATE VIEW seznam_zbozi AS
    select z.nazev as 'Zboží', z.pn as 'Produktové číslo', k.kategorie as 'Kategorie',
    v.vyrobce as 'Výrobce', mj.zkratka as 'Měrná jednotka' from zbozi as z
    join kategorie as k
    on z.id_kat = k.id_kat
    join vyrobce as v
    on z.id_vyr = v.id_vyr
    join merna_jednotka as mj
    on z.id_mj = mj.id_mj;
```

Dotaz na vypsání dat z pohledu: `SELECT * FROM seznam_zbozi;`

```
CREATE VIEW seznam_spolecnosti AS
    select f.nazev as 'Společnost', k.telefon as 'Telefon', k.email as 'Email', f.mesto as
    'Město', f.ulice as 'Ulice', f.cp as 'Č.p.', f.psc as 'PSČ', f.stat as 'Stát'
    from firma as f
    join kontakt_firmy as k
    on f.id_firma = k.id_firma;

SELECT * FROM seznam_spolecnosti;

CREATE VIEW seznam_zamestnancu AS
    select u.prijmeni as 'Příjmení', u.jmeno as 'Jméno', k.telefon as 'Telefon', k.email
    as 'Email', u.mesto as 'Město', u.ulice as 'Ulice', u.cp as 'Č.p.', u.psc as 'PSČ',
    u.stat as 'Stát'
    from uzivatel as u
    join kontakt_uzivatele as k
    on u.id_uziv = k.id_uziv;

SELECT * FROM seznam_zamestnancu;

CREATE VIEW seznam_dodavatelu AS
    select f.nazev as 'Dodavatel', sum(pp.cena_nak) as 'Celkový příjem', kf.telefon
    as 'Telefon', kf.email as 'Email', f.mesto as 'Město'
    from prijemka as p
    join polozka_prijemky as pp
    on p.id_pr = pp.id_pr
    join firma as f
    on p.id_dod = f.id_firma
    join kontakt_firmy as kf
    on f.id_firma = kf.id_firma
    group by f.nazev;

SELECT * FROM seznam_dodavatelu;
```

```
CREATE VIEW seznam_odberatelu AS
    select f.nazev as 'Odberatel', sum(pv.cena_p) as 'Celkový výdej', kf.telefon as
    'Telefon', kf.email as 'Email', f.mesto as 'Město'
    from vydejka as v
    join polozka_vydejka as pv
    on v.id_vyd = pv.id_vyd
    join firma as f
    on v.id_odb = f.id_firma
    join kontakt_firmy as kf
    on f.id_firma = kf.id_firma
    group by f.nazev;

SELECT * FROM seznam_odberatelu;
```

4.3.4 Procedura Vlož uživatele

Tato procedura slouží k vložení nového uživatele a jeho kontaktu do databáze. Po zadání vstupních parametrů procedura ověří, zda se uživatel již v databázi vyskytuje. V tom případě bude výstupem ID již existujícího uživatele. V případě, že je zadáván nový uživatel, budou vloženy zadané atributy tohoto uživatele do tabulek *Uzivatel* a *Kontakt_uzivatele* a výstupem bude ID právě vloženého uživatele.

```
CREATE PROCEDURE vloz_uzivatele(
jmeno1 varchar(20), prijmeni1 varchar(30), rc1 char(10), ulice1 varchar(25), cp1
varchar(10), psc1 char(5), mesto1 varchar(30), stat1 varchar(20), opravneni1 int,
telefon1 char(9),
email1 varchar(45)
)
BEGIN
declare nove_id int;
set nove_id = 0;
set nove_id = (select id_uziv from uzivatel where prijmeni = prijmeni1 and rc = rc1);
if (nove_id is not null) then
    select nove_id as 'Uživatel se v databázi již vyskytuje s tímto ID:';
```

```
elseif (nove_id is null) then
        insert into uživatel (jmeno, prijmeni, rc, ulice, cp, psc, mesto, stat, id_opr)
        values (jmeno1, prijmeni1, rc1, ulice1, cp1, psc1, mesto1, stat1, opravneni1);
        set nove_id = @@identity;
        insert into kontakt_uzivatele (telefon, email,  id_uziv)
        values (telefon1, email1, nove_id);
        select nove_id as 'ID nového uživatele je:';
end if;
END
```

Procedura je volána (spuštěna) následujícím dotazem:

```
CALL vloz_uzivatele('Martin', 'Veselý', '8305224654', 'Veveří', '39', '61600', 'Brno',
'ČR', 2, '723456789', 'martinvesely@seznam.cz');
```

4.3.5 Procedura Vlož firmu

Díky této proceduře je zjednodušeno vložení nového dodavatele či odběratele a jeho kontaktu do databáze. Funguje na stejném principu jako procedura *vloz_uzivatele*. Nejprve se ověří, zda zadávaná firma již v databázi existuje. Pokud ano, výstupem bude ID již existující firmy. Pokud je zadávaná nová firma, budou vloženy zadané atributy této firmy do tabulek *firma* a *kontakt_firmy* a výstupem bude ID právě vložené firmy.

```
CREATE PROCEDURE `vloz_firmu`(
`nazev1` varchar(30), `ulice1` varchar(25), `cp1` varchar(10), `psc1` char(5), `mesto1`
varchar(30), `stat1` varchar(20), `telefon1` char(9), `email1` varchar(45), `ico1`
varchar(8), `dic1` varchar(10)
)
BEGIN
declare new_id_firma int;
set new_id_firma = 0;
set new_id_firma = (select id_firma from firma where nazev = nazev1 and ico = ico1);

if (new_id_firma is not null) then
```

```
        select new_id_firma as 'Firma se v databázi již vyskytuje s tímto ID:';
elseif (new_id_firma is null) then
        insert into firma (nazev, ico, dic, ulice, cp, psc, mesto, stat)
        values (nazev1, ico1, dic1, ulice1, cp1, psc1, mesto1, stat1);
        set new_id_firma = @@identity;
        insert into kontakt_firmy (telefon, email, id_firma)
        values (telefon1, email1, new_id_firma);
        select new_id_firma as 'ID nové firmy je:';
end if;
END
```

Dotaz na spuštění procedury:

```
CALL vloz_firmu ('Cooler s.r.o.', 'Jánská',  '300', '60200', 'Brno', 'ČR', '736449367',
'info@cooler.cz', '12345678', 'CZ12345678')
```

4.3.6 Procedura Vlož zboží

Tato procedura umožňuje vložení nového zboží do databáze. Po zadání vstupních parametrů je otestováno, zda se pod zadaným označením výrobce, kategorie a měrné jednotky v databázi vyskytují záznamy a pokud ne, tak je na to uživatel upozorněn. V případě, že jsou vstupy korektní, procedura vloží informace o zboží do tabulky *zbozi* spolu s provázáním na tabulky *vyrobce*, *kategorie* a *merna_jednotka*. Výstupem bude ID právě vloženého zboží.

```
CREATE PROCEDURE vloz_zbozi(
nazev1 varchar(45),
popis1 mediumtext,
pn1 varchar(20),
zar_doba1 int,
vyrobce1 varchar(30),
kategorie1 varchar(30),
mj1 varchar(30)
)
BEGIN
```

```
declare id_vyr1 int;
declare id_kat1 int;
declare id_mj1 int;
declare nove_id_zb int;

set id_vyr1 = 0;
set id_kat1 = 0;
set id_mj1 = 0;
set nove_id_zb = 0;

set id_vyr1 = (select id_vyr from vyrobce where vyrobce = vyrobce1);
set id_kat1 = (select id_kat from kategorie where kategorie = kategorie1);
set id_mj1 = (select id_mj from merna_jednotka where (mj = mj1 or zkratka = mj1));
set nove_id_zb = (select id_zb from zbozi where nazev = nazev1 and pn = pn1);

if (id_vyr1 is null) then
        select id_vyr1 as 'Pod zadaným označením se v databázi nevyskytuje žádný
výrobce';
elseif (id_kat1 is null) then
        select id_kat1 as 'Pod zadaným označením v databázi nevyskytuje žádná
kategorie';
elseif (id_mj1 is null) then
        select id_mj1 as 'Pod zadaným označením se v databázi nevyskytuje žádná
měrná jednotka';
elseif (id_vyr1 is not null) and (id_kat1 is not null) and (id_mj1 is not null) then
        if (nove_id_zb is not null) then
                select nove_id_zb as 'Zboží se v databázi již vyskytuje s tímto ID:';
        elseif (nove_id_zb is null) then
                set nove_id_zb = @@identity+1;
                insert into zbozi (nazev, popis, pn, zar_doba, id_vyr, id_kat, id_mj)
                values (nazev1, popis1, pn1, zar_doba1, id_vyr1, id_kat1, id_mj1);
                select nove_id_zb as 'ID nového zboží je:';
```

```
        end if;
end if;
END
```

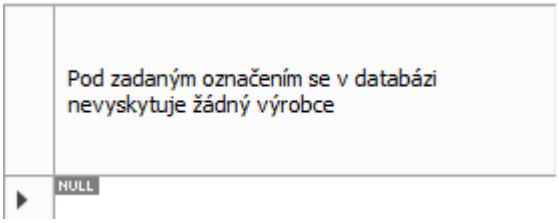
Spuštění procedury:

```
CALL vloz_zbozi ('WD Caviar Blue EX - 1TB', 'Rychlý pevný disk s vysokou kapacitou 1 TB, 64 MB vyrovnávací paměť, 7200 ot/min, rozhraní SATA 6Gb/s.', 'WD10EZEX', 2, 'WD', 'HDD', 'ks');
```

Příklad zadání zboží s neexistujícím výrobcem Nike:

```
CALL vloz_zbozi ('Seagate Barracuda 7200.14 - 1TB', 'Interní 3.5" disk, kapacita 1 TB, 7200 otáček/min., 64 MB cache, moderní rozhraní SATA 6Gb/s (SATA 3.0), funkce NCQ - Native Command Queuing.', 'ST1000DM003', 2, 'Nike', 'HDD', 'ks');
```

Výstup:



Obrázek č. 20: Výstup procedury Vlož zboží

4.3.7 Procedura Celkový příjem

Tato procedura slouží k vypočítání celkové částky za příjem zboží za dané časové období. Vstupními parametry procedury jsou *datum_od* a *datum_do*, které ohraničují konkrétní požadované období. Výstupní parametr je jeden a to *vysledek*, který vrací vypočtenou celkovou částku za příjem zboží. Při spuštění procedury je u tohoto parametru nutné uvést jako prefix jeho názvu symbol @, aby mohl být parametr znovu volán vně procedury. Je zde využita agregační funkce SUM(), která zjistí součet položek v právě seskupovaných řádcích (v tomto případě součet nákupních cen).

```
CREATE PROCEDURE celkovy_prijem(
datum_od date,
datum_do date,
out vysledek int
)
BEGIN
set vysledek = (select sum(pp.cena_nak) from polozka_prijemky as pp
                join prijemka as p
                on p.id_pr = pp.id_pr
                and p.datum >= datum_od
                and p.datum <= datum_do);
select vysledek as 'Celková částka za příjem zboží za dané období je:';
END
```

Spuštění procedury:

```
CALL celkovy_prijem("2015-05-01", "2015-05-31", @vysledek);
```

Výstup:

	Celková částka za příjem zboží za dané období je:
▶	21509

Obrázek č. 21: Výstup procedury Celkový příjem

4.3.8 Procedura Celkový výdej

Díky této proceduře je možné vypočítat celkovou částku za výdej zboží za dané časové období. Funguje na stejném principu jak procedura *Celkový výdej* s tím rozdílem, že se spojují tabulky *Položka výdejky* a *Výdejka*.

```
CREATE PROCEDURE celkovy_vydej(
datum_od date,
datum_do date,
out vysledek int
)
BEGIN
set vysledek = (select sum(pv.cena_p) from polozka_vydejky as pv
                join vydejka as v
                on v.id_vyd = pv.id_vyd
                and v.datum >= datum_od
                and v.datum <= datum_do);
select vysledek as 'Celková částka za výdej zboží za dané období je:';
END
```

Spuštění procedury:

```
CALL celkovy_vydej("2015-05-01", "2015-05-31", @vysledek);
```

Výstup:

	Celková částka za výdej zboží za dané období je:
▶	25811

Obrázek č. 22: Výstup procedury Celkový výdej

4.4 Zhodnocení přínosů

Zpracováním návrhu databáze skladové evidence byla vytvořena databázová struktura na základě požadavků zadavatele. Tato databázová struktura bude následně využita pro vývoj aplikačního prostředí.

Databáze umožňuje uchovávání informací o zboží, jeho příjmu na sklad, výdeje ze skladu, dodavatelích a odběratelích společnosti, zaměstnancích a především pohybu zboží v rámci společnosti, díky čemuž ve společnosti nebude potřeba dvojího vedení skladové evidence a tím pádem bude celkově zefektivněna práce se všemi evidovanými daty.

Hlavní přínos práce tedy spočívá v navržení databázové struktury, na základě které bude databáze reálně použita v aplikaci, která bude uplatněna v praxi.

4.4.1 Ekonomický pohled

Společnost disponuje vlastním databázovým serverem MySQL, který běží na operačním systému Linux. V tomto ohledu firma nemusí pro chod databáze vynakládat žádné finanční prostředky na pořízení softwaru.

Vzhledem k tomu, že většinou firmy vytvářejí již hotová řešení společně s aplikačním prostředím, budou náklady na tvorbu databáze odhadnuty.

Pracnost vytvoření všech částí návrhu databáze je odhadnuta na dobu 2 týdnů při úvazku 5 člověkodnů týdně (1 člověkoden = 8 člověkohodin). Odhad celkové doby tedy představuje 80 hodin čisté práce. Při hodinové mzdě 200 Kč jsou celkové náklady 16000 Kč. Délka zpracování návrhu zahrnuje všechny částí návrhu databáze včetně počáteční analýzy, tvorby konceptuálního, logického i fyzického návrhu.

Ekonomický přínos práce spočívá v ušetření nákladů na vytvoření návrhu databázové struktury, které odhaduje níže uvedená tabulka.

	Odhadovaná pracnost (člověkohodiny)	**Hodinová mzda (Kč)**	**Náklady (Kč)**
Návrh databáze	80	200	16000

Tabulka č. 3: Odhadované náklady na realizaci návrhu

ZÁVĚR

Cílem bakalářské práce bylo vytvořit návrh databáze skladového hospodářství ve společnosti DATEL PLUS s.r.o.

V první teoretické části byla představena problematika databází od základních pojmů přes fáze návrhu databáze, relační datový model a na něm založené relační databáze, normalizaci, až po jazyk SQL a jeho jednotlivé příkazy a datové typy.

Druhá analytická část se zaměřila na popis společnosti a analýzu jejího současného vedení skladové evidence, na základě které byly stanoveny požadavky na databázi. Taktéž byla provedena analýza a výběr vhodné databázové platformy.

Ve třetí části práce byl vytvořen vlastní návrh řešení představující návrh databáze, skládající se ze tří dílčí částí a to konceptuálního, logického a fyzického návrhu. V konceptuálním návrhu byly identifikovány důležité objekty, o kterých bylo v databázi nutné uchovávat informace, a vztahy mezi těmito objekty. Ve fázi logického návrhu vznikl logický model dat, který reprezentuje vytvořené normalizované tabulky. Fyzický návrh představoval implementaci do cílového DBMS pomocí jazyk SQL.

Tím byl stanovený cíl práce splněn. Navržené databázové struktury budou ve společnosti využity pro následný vývoj aplikačního prostředí.

SEZNAM POUŽITÝCH ZDROJŮ

[1] STEPHENS, R. K. a R. R. PLEW. *Naučte se SQL za 21 dní.* Brno: Computer Press, 2004. 491 s. ISBN 80-722-6870-8.

[2] CONOLLY, T., C. BEGG a R. HOLOWCZAK. *Mistrovství - databáze: Profesionální průvodce tvorbou efektivních databází.* Praha: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.

[3] KOCH, M. *Datové a funkční modelování*. 2. vyd. Brno: Akademické nakladatelství CERM, 2006. 108 s. ISBN 80-214-2724-8.

[4] LACKO, L. *1001 tipů a triků pro SQL.* Brno: Computer Press, 2011. 416 s. ISBN 978-80-251-3010-0.

[5] ELMASRI, R. a S. B. NAVATHE. *Fundamentals of Database Systems*. 6. vyd. Boston: Addison Wesley, 2010. 1172 s. ISBN 978-0-136-08620-8.

[6] OPPEL, A. *Databáze bez předchozích znalostí*. Brno: Computer Press, 2006. 316 s. ISBN 80-251-1199-7.

[7] OPPEL, A. *SQL bez předchozích znalostí.* Brno: Computer Press, 2008. 240 s. ISBN 978-80-251-1707-1.

[8] KROENKE, D. M. a D. J. AUER, *Databáze.* Brno: Computer Press, 2015. 496 s. ISBN 978-80-251-4352-0

[9] HERNANDEZ M. J. *Návrh databází.* Praha: Grada, 2006. 408 s. ISBN 80-247-0900-7.

[10] WELLING, L. a L. THOMSON. *MySQL: průvodce základy databázového systému.* Brno: CP Books, 2005. 255 s. ISBN 80-251-0671-3.

[11] PROCHÁZKA, D. *Oracle: průvodce správou, využitím a programováním nad databázovým systémem.* Praha: Grada, 2009. 168 s. ISBN 978-80-247-2762-2.

[12] Pohoda – účetní program. *Pohoda – ekonomický a informační systém* [online]. 2015 [cit. 2015-05-20]. Dostupné z: http://www.stormware.cz/pohoda/

[13] Skladové hospodářství. *Pohoda – ekonomický a informační systém* [online]. 2015 [cit. 2015-05-21]. Dostupné z: http://www.stormware.cz/pohoda/sklady.aspx

[14] Database 12c. *Oracle.com* [online]. 2015 [cit. 2015-05-21]. Dostupné z: http://www.oracle.com/cz/database/overview/index.html

[15] SQL Server 2014. *Microsoft: Microsoft SQL Server* [online]. 2015 [cit. 2015-05-21]. Dostupné z: http://www.microsoft.com/cs-cz/server-cloud/products/sql-server/

[16] Edice systému SQL Server. *Microsoft: Microsoft SQL Server* [online]. 2015 [cit. 2015-05-21]. Dostupné z: http://www.microsoft.com/cs-cz/server-cloud/products/sql-server-editions/

[17] Why MySQL?. *MySQL.com* [online]. 2015 [cit. 2015-05-21]. Dostupné z: https://www.mysql.com/why-mysql/

[18] Products. *MySQL.com* [online]. 2015 [cit. 2015-05-21]. Dostupné z: http://www.mysql.com/products/

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM POJMŮ A ZKRATEK

Pojem	Zkratka	Význam [zdroj]
Database Management System	DBMS	Softwarový systém, pomocí něhož je možné definovat, vytvářet a udržovat databázi a poskytovat řízený přístup k této databázi. [2]
Entitně-relační diagram	ERD	Podrobná reprezentace entit, atributů a relací. [2]
Entita		Množina objektů se shodnými vlastnostmi, které uživatel nebo organizace identifikuje jako nezávisle existující objekty. [2]
Relace		1. tabulka se sloupci a řádky v relačním modelu 2. (ve smyslu *vztahu* entit) množina smysluplných spojení mezi entitami. [2]
Atribut		Hodnota, která určuje některou podstatnou vlastnost entity či vztahu. [4]
Doména		Množina přípustných hodnot pro jeden či více atributů. [2]
Klíč		Atribut nebo skupina atributů (sloupec nebo skupina sloupců v relačním modelu), které jedinečně identifikují každý výskyt entity. [2]
Primární klíč	PK	Klíč, který je zvolen pro jedinečnou identifikaci příslušné entity, respektive pro jedinečné určení záznamu v tabulce. [2]
Cizí klíč	FK	Sloupec nebo skupina sloupců v jediné tabulce, která jsou propojeny na primární klíč v jiné tabulce. [4]
Kandidátní klíč		Klíč, který obsahuje jen minimální počet sloupců nutných k jedinečné identifikaci záznamů. [2]
Alternativní klíč		Kandidátní klíč, který není zvolen jako primární klíč. [2]
Normalizace		Technika používaná pro vytvoření sady tabulek s minimální redundancí. [2]

Normální formy	1NF, 2NF, 3NF	Fáze procesu normalizace. První tři normální formy se nazývají první normální forma (1NF), druhá normální forma (2NF) a třetí normální forma (3NF). [2]
Redundantní data		Duplikovaná data, která jsou uložena ve více než jedné tabulce. [2]
Integrita dat		Označuje správnost a konzistenci uložených dat. [2]
Integritní omezení		Pravidla pro definování nebo omezení některých vlastností dat užívaných organizací. [2]
Konzistence dat		Schopnost v kterýkoliv okamžik vyjadřovat nějakou (možnou) informaci o reálném světě. [4]
Podkladová tabulka		Pojmenovaná tabulka s fyzicky uloženými záznamy v databázi. [2]
Null	NULL	Reprezentuje chybějící nebo neznámou hodnotu, nereprezentuje nulu nebo prázdný textový řetězec. [9]
Structured Query Language	SQL	*Strukturovaný dotazovací jazyk* (SQL) je neprocedurální databázový jazyk pro relační systémy řízení báze dat, používaný k vytváření, udržování, úpravám a kladení dotazů relační databázi. [9]
Open source		Software s volně dostupnými zdrojovými kódy
Pohled		„Virtuální tabulka“, která v databázi nemusí existovat, ale generuje ji DBMS z tabulek, na kterých je pohled založen, kdykoli se k němu přistupuje. [4]
Procedura		Podprogram, který může přijímat množinu parametrů předávaných mu při volání a provádět řadu činností. [2]

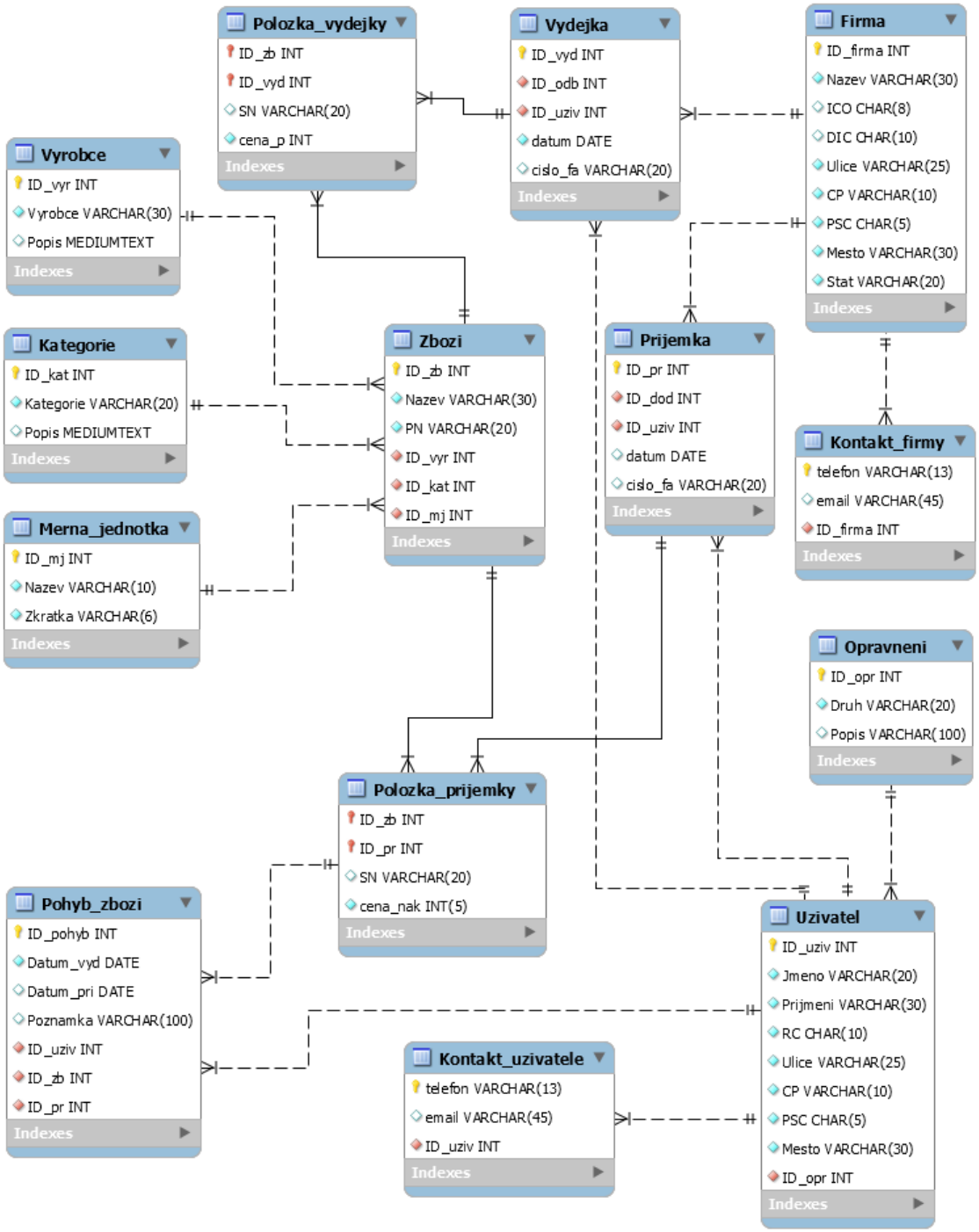
SEZNAM PŘÍLOH

Příloha č. 1: Datový slovník

Tabulka	Položka	Typ	Délka	Klíč	Omezení
Výrobce	ID_vyr	int		PK	not null
	Vyrobce	varchar	30		not null
	Popis	mediumtext			
Kategorie	ID_kat	int		PK	not null
	Kategorie	varchar	20		not null
	Popis	mediumtext			
Měrná jednotka	ID_mj	int		PK	not null
	Nazev	varchar	10		not null
	Zkratka	varchar	6		not null
Zboží	ID_zb	int		PK	not null
	Nazev	varchar	30		not null
	PN	varchar	20		not null
	Zarucni_doba	int	1		not null
	ID_kat	int		FK	not null
	ID_vyr	int		FK	not null
	ID_mj	int		FK	not null
Firma	ID_firma	int		PK	not null
	Nazev	varchar	30		not null
	ICO	char	8		
	DIC	char	10		
	Ulice	varchar	25		not null
	CP	varchar	10		not null
	PSC	char	5		not null
	Mesto	varchar	30		not null
	Stat	varchar	20		not null
Kontakt firmy	Telefon	varchar	13	PK	not null
	Email	varchar	45		
	ID_firma	int		FK	not null
Oprávnění	ID_opravneni	int		PK	not null
	Druh	varchar	20		not null
	Popis	varchar	100		
Uživatel	ID_uziv	int		PK	not null
	ID_opravneni	int		FK	not null
	Jmeno	varchar	20		not null
	Prijmeni	varchar	30		not null
	RC	char	10		not null
	Ulice	varchar	25		not null
	CP	varchar	10		not null
	PSC	char	5		not null
	Mesto	varchar	30		not null
	Stat	varchar	20		not null

Kontakt uživatele	Telefon	varchar	13	PK	not null
	Email	varchar	45		
	ID_uziv	int		FK	not null
Příjemka	ID_pr	int		PK	not null
	ID_dod	int		FK	not null
	ID_uziv	int		FK	not null
	Datum	date			not null
	Cislo_fa	varchar	20		
Položka příjemky	ID_zb	int		PK	not null
	ID_pr	int		PK	not null
	SN	varchar	20		
	cena_nak	int			not null
Výdejka	ID_vyd	int		PK	not null
	ID_odb	int		FK	not null
	ID_uziv	int		FK	not null
	Datum	date			not null
	Cislo_fa	varchar	20		
Položka výdejky	ID_zb	int		PK	not null
	ID_vyd	int		PK	not null
	SN	varchar	20		
	cena_p	int			not null
Pohyb zboží	ID_pohyb	int		PK	not null
	Datum_vyd	date			not null
	Datum_pri	date			not null
	Poznamka	varchar	100		
	ID_uziv	int		FK	not null
	ID_zb	int		FK	not null
	ID_pr	int		FK	not null

Příloha č. 2: ER diagram



Příloha č. 3: Zdrojový kód

```
CREATE TABLE IF NOT EXISTS vyrobce (
  id_vyr int NOT NULL AUTO_INCREMENT,
  vyrobce varchar(30) NOT NULL,
  popis varchar(100),
  PRIMARY KEY (id_vyr)
);

CREATE TABLE IF NOT EXISTS kategorie (
  id_kat int NOT NULL AUTO_INCREMENT,
  kategorie varchar(30) NOT NULL,
  popis varchar(100),
  PRIMARY KEY (id_kat)
);

CREATE TABLE IF NOT EXISTS merna_jednotka (
  id_mj int NOT NULL AUTO_INCREMENT,
  nazev varchar(10) NOT NULL,
  zkratka varchar(6) NOT NULL,
  PRIMARY KEY (id_mj)
);

CREATE TABLE IF NOT EXISTS zbozi (
  id_zb int NOT NULL AUTO_INCREMENT,
  id_vyr int NOT NULL,
  id_kat int NOT NULL,
```

```
  id_mj int NOT NULL,
  nazev varchar(30) NOT NULL,
  popis varchar(100),
  pn varchar(20) NOT NULL,
  zar_doba int NOT NULL,
  PRIMARY KEY (id_zb),
  FOREIGN KEY (id_vyr) REFERENCES vyrobce (id_vyr),
  FOREIGN KEY (id_kat) REFERENCES kategorie (id_kat),
  FOREIGN KEY (id_mj) REFERENCES merna_jednotka (id_mj)
);

CREATE TABLE IF NOT EXISTS firma (
  id_firma int NOT NULL AUTO_INCREMENT,
  nazev varchar(30) NOT NULL,
  ico varchar(8),
  dic varchar(10),
  ulice varchar(25) NOT NULL,
  cp varchar(10) NOT NULL,
  psc varchar(5) NOT NULL,
  mesto varchar(30) NOT NULL,
  stat varchar(20) NOT NULL,
  PRIMARY KEY (id_firma)
);

CREATE TABLE IF NOT EXISTS kontakt_firmy (
  telefon char(9) NOT NULL,
```

```
  email varchar(45),
  id_firma int NOT NULL,
  PRIMARY KEY (telefon),
  FOREIGN KEY (id_firma) REFERENCES firma (id_firma)
  );

CREATE TABLE IF NOT EXISTS opravneni (
  id_opr int NOT NULL AUTO_INCREMENT,
  druh varchar(20) NOT NULL,
  popis varchar(100),
  PRIMARY KEY (id_opr)
);

CREATE TABLE IF NOT EXISTS uzivatel (
  id_uziv int NOT NULL AUTO_INCREMENT,
  jmeno varchar(20) NOT NULL,
  prijmeni varchar(30) NOT NULL,
  rc char(10) NOT NULL,
  ulice varchar(25) NOT NULL,
  cp varchar(10) NOT NULL,
  psc varchar(5) NOT NULL,
  mesto varchar(30) NOT NULL,
  stat varchar(20) NOT NULL,
  id_opr int NOT NULL,
  PRIMARY KEY (id_uziv),
  FOREIGN KEY (id_opr) REFERENCES opravneni (id_opr)
```

```
);

CREATE TABLE IF NOT EXISTS kontakt_uzivatele (
  telefon char(9) NOT NULL,
  email varchar(45),
  id_uziv int NOT NULL,
  PRIMARY KEY (telefon),
  FOREIGN KEY (id_uziv) REFERENCES uzivatel (id_uziv)
  );

CREATE TABLE IF NOT EXISTS prijemka (
  id_pr int NOT NULL AUTO_INCREMENT,
  datum date NOT NULL,
  cis_fa varchar(20),
  id_dod int NOT NULL,
  id_uziv int NOT NULL,
  PRIMARY KEY (id_pr),
  FOREIGN KEY (id_dod) REFERENCES firma (id_firma),
  FOREIGN KEY (id_uziv) REFERENCES uzivatel (id_uziv)
);

CREATE TABLE IF NOT EXISTS polozka_prijemky (
  id_pr int NOT NULL REFERENCES příjemka (id_pr),
  id_zb int NOT NULL REFERENCES zbozi (id_zb),
  sn varchar(20),
  cena_nak int NOT NULL,
```

```
  PRIMARY KEY (id_pr, id_zb)
);

CREATE TABLE IF NOT EXISTS vydejka (
  id_vyd int NOT NULL AUTO_INCREMENT,
  datum date NOT NULL,
  cis_fa varchar(20),
  id_odb int NOT NULL,
  id_uziv int NOT NULL,
  PRIMARY KEY (id_vyd),
  FOREIGN KEY (id_odb) REFERENCES firma (id_firma),
  FOREIGN KEY (id_uziv) REFERENCES uzivatel (id_uziv)
);

CREATE TABLE IF NOT EXISTS polozka_vydejky (
  id_vyd int NOT NULL REFERENCES vydejka (id_vyd),
  id_zb int NOT NULL REFERENCES zbozi (id_zb),
  sn varchar(20),
  cena_p int NOT NULL,
  PRIMARY KEY (id_vyd, id_zb)
);

CREATE TABLE IF NOT EXISTS pohyb_zbozi (
  id_pohyb int NOT NULL AUTO_INCREMENT,
  datum_vyd datetime NOT NULL,
  datum_pri datetime,
```

```
poznamka varchar(50),
id_uziv int NOT NULL,
id_zb int NOT NULL,
id_pr int NOT NULL,
PRIMARY KEY (id_pohyb),
FOREIGN KEY (id_uziv) REFERENCES uzivatel (id_uziv)
FOREIGN KEY (id_zb, id_pr) REFERENCES polozka_prijemky (id_zb, id_pr)
);
```