Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Tvorba datového modelu vybrané agendy ekonomického informačního systému podniku

Vypracoval: Jiří Boháč
Vedoucí práce: Ing. Petr Hanzal Ph.D.

České Budějovice 2018

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta ekonomická
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jiří BOHÁČ**
Osobní číslo:	**E14365**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Tvorba datového modelu vybrané agendy ekonomického informačního systému podniku**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Cílem práce je analýza vybrané agendy podniku, tvorba konceptuálního a fyzického datového modelu informačního systému pro tuto agendu. Praktická část se bude zabývat realizací tohoto datového modelu na vybraném databázovém systému.

Metodický postup:

1. V prvotní fázi práce je nutné provést analýzu vybrané podnikové agendy.
2. Dále je nutné určit prvky konceptuálního modelu agendy a tento model vytvořit v dostupném softwarovém nástroji pro tvorbu konceptuálních modelů.
3. Dále je nutné provést normalizaci dat.
4. V další fázi je třeba konceptuální model převést do fyzického modelu a implementovat celý datový model na vybraném databázovém stroji. Práce může být rozvinuta i do jednoduché aplikace simulující základní funkce agendy.
5. Závěr a doporučení.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **50 - 60 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **Conolly T., Begg C., & Holowczak, R. (2009).** *Mistrovství - Databáze: Profesionální průvodce tvorbou efektivních databází.* **Brno: Computer Press.**
2. **Hernandez, M. J. (2006).** *Návrh databází.* **Praha: Grada.**
3. **Kanisová, H., & Müller, M. (2006).** *UML srozumitelně.* **Praha: Computer Press.**
4. **Šimůnek, M. (2001).** *SQL - kompletní kapesní průvodce.* **Praha: Grada.**

Vedoucí bakalářské práce: **Ing. Petr Hanzal, Ph.D.**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **18. října 2017**

Termín odevzdání bakalářské práce: **13. dubna 2018**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

L.S.

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

RNDr. Jana Klicnarová, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 18. října 2017

Prohlášení

Prohlašuji, že svoji bakalářskou práci na téma „Tvorba datového modelu vybrané agendy ekonomického informačního systému podniku“ jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to – v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

..…… ……..

Datum Podpis

Poděkování

Chtěl bych velmi poděkovat Ing. Petru Hanzalovi Ph.D. za cenné rady a odborné vedení celé bakalářské práce.

Obsah

Úvod

S rozvojem středních i velkých podniků, inovacemi v managementu jako takovém a společně se změnami v pojetí ekonomického úseku, přišly i změny v informačních systémech jako takových.

Nákladnost dokonalejší verze informačních systémů však směřuje přirozeně podniky k tomu, aby se zajímaly i o samotnou počáteční investici za takový systém, obdobně jako rychlý vývoj IT techniky by je měl vést k tomu, aby pozornost při jeho výběru věnovaly i možnostem jeho rozšíření.

Pod ekonomickou agendu, zejména velkých organizací, je toho dnes zařazeno mnohem více a dřívější a samostatné úseky se tak přiřadily k původně samostatným agendám účetním a podobně.

Sama tvorba informačního systému obnáší řadu fází, které musí být kvalitně realizovány od samotné tvorby zadání, implementaci až po testovací fázi v praxi.

První kapitola práce seznamuje čtenáře s pojetím informačního systému a jeho komponentami, detailnějším rozdělením ekonomického softwaru.

Druhá kapitola je věnována tvorbě skladového informačního systému. V první fázi návrhem pomocí programu Toad Data Modeler od společnosti Quest Software a následným vytvořením aplikace v programu Access od firmy Microsoft.

Cíl

Cílem bakalářské práce je čtenáře seznámit s firemními informačními systémy, které zabezpečují sběr, přenos, zpracování a uchování dat v podniku. Čtenáři budou postupně obeznámeni se základním rozdělením informačního systému na komponenty, podle jejich určení.

Dále se dozví, jaké prvky a kroky je potřeba vykonat před a při tvorbě databázové aplikace. Závěrem bude zobrazena sama tvorba celé aplikace od tvorby konceptuálního modelu v programu Visio, logického modelu v Toad Data Modeleru a samotné tvorby v programu Access. Bude zobrazeno propojování tabulek, klíčování a budou zobrazeny rozdíly mezi jednotlivými normalizačními formami.

1 Firemní informační systémy

V současnosti se lze setkat s nemalým množstvím vymezení informačního systému. Molnár (2000) ve své publikaci definuje informační systém jako soubor jedinců, dále technických prostředků i metod (tj. programů), které zabezpečující sběr, přenos, zpracování i uchování dat za záměrem prezentovat informace podle potřeb uživatelů, kteří jsou sami činní v systémech řízení.

Informační systém je přitom složen z několika komponent:

- Technických prostředků (hardware)
- Programových prostředků (software)
- Organizačních prostředků (orgware) – souboru pravidel vymezujících provoz i užití informačního systému a také informačních technologií
- Lidského faktoru (peopleware) – jedná se o způsob adaptace jedince v počítačovém prostředí
- Vlivů reálného světa (jako např. informační zdroje, normy a legislativní nařízení) – týkající se kontextu informačního systému (Tvrdíková, 2000).

Z uvedených komponent výše vyplývá, že aby byl takový systém správně nasazen a využit v daném podniku, je třeba tyto faktory vnímat komplexně. Pravidlem přitom zůstává, že informačnímu systému je třeba porozumět jak z technologického pohledu, tak z pohledu skupin jeho uživatelů i z hlediska procesního uspořádání dané firmy (Basl, 2002).

Podle zaměření lze informační systémy zařadit obecně do několika kategorií, jak poukáže text níže.

1.1 Členění podnikových informačních systémů

Podnikové informační systémy se dělí na tři základní obecné skupiny:

- Účetní software
- Ekonomický software
- ERP (Enterprise Resource Planning), (Epadus.cz, 2009).

1.2 Účetní software

Účetní software lze popsat jako software pokrývající obvykle podvojné účetnictví a také daňovou evidenci. V současnosti již však klasický účetní software na trhu nelze nalézt, jelikož jen daňová evidence, anebo podvojné účetnictví, je dnes samostatně užíváno málo (Epadus.cz, 2009). Díky historickému vývoji již dnešní firmy řadí tyto úkony pod ekonomické úseky jako takové.

1.3 Ekonomický software

Kromě podvojného účetnictví a daňové evidence tedy obsahuje ekonomický software i jiné agendy, které slouží k uspokojení potřeb daného podnikatele. Adresář je doplněný grafy i tzv. přehledy dokladů tříděných podle jednotlivých obchodních partnerů. Může také obsahovat skladové hospodářství, knihu jízd, automatizaci objednávání zásob, evidenci dokumentů týkajících se obchodních partnerů, odeslané i přijaté objednávky. Tradičními představiteli jsou např. programy POHODA nebo Money S3 (Epadus.cz, 2009). Ty však nepodporují ERP koncepci v pojetí procesně orientované podnikové strategie (Sodomka, 2010).

1.3.1 Rozdělení ekonomického softwaru

Český trh nabízí tyto typy informačních systémů ekonomicky orientovaných:

- Morálně i technicky zastaralé systémy na platformě DOS – na trhu zůstaly ovšem z toho důvodu, že určitým firmám i tak stačí. Charakterizuje je omezená intuitivnost i ergonomie uživatelského prostředí (Epadus.cz, 2009). Jako příklad lze jmenovat program ORAX – DOS (Orax.cz, 2018) či program Amikon (Amikon.cz, „Nedatováno“).
- Jednoduché systémy na platformě Windows – své uplatnění nalezly v oblasti ekonomiky řady firem, jsou ovšem limitované v možnosti podpořit její růst. Klasickým zástupcem je program zvaný Pohoda nebo Stereo (Epadus.cz, 2009).

- Pokročilejší ekonomické systémy – již podporují růst firmy do budoucnosti. Podporují oblasti logistiky, výroby, CRM i analýz pro manažery (Epadus.cz, 2009). Jako příklad lze uvést program Abra G2 nebo Money S3 (Sodomka, 2010).

1.4 ERP systémy

ERP systémy již patří k vyšší formě ekonomického softwaru. Od tradičního softwaru se liší modularitou i řešením „na míru“. Využití mají jak v oblasti účetnictví, tak v napomáhání splnit cíle firmy. Z toho důvodu jejich vlastní instalaci předchází jednání s klientem pro identifikování jeho potřeb. Nevýhodou je však vyšší pořizovací cena nebo složitější implementace. Ovšem dobře implementovaný ERP systém se může stát z hlediska funkčnosti nebo efektivity provozu nesrovnatelně přínosným v porovnání se standardním ekonomickým softwarem. Jeho smyslem je přizpůsobit se potřebám klienta na maximum na rozdíl od ekonomických systémů, kde se zákazník přizpůsobuje naopak ekonomickému systému. K zástupcům patří například programy Microsoft Dynamics NAV, Helios Orange, Money S5, Altus Vario nebo ABRA G4, SAP, BMD (Sodomka, 2012).

1.4.1 Klasifikace ERP systémů podle oborového a funkčního zaměření

ERP systémy se třídí podle toho, jak zvládnou pokrýt čtyři klíčové podnikové interní procesy. K těm je řazena personalistika, výroba, (vnitřní) logistika a ekonomika:

- All-in-One – jedná se o systémy mající schopnost pokrýt všechny klíčové interní procesy. Typická je pro ně vysoká úroveň integrace, která stačí většině organizací, ovšem jejich nevýhodou je nákladnější úprava na míru i nižší detailní funkcionalita.
- Best-of-Breed – jedná se o systémy zaměřující se na specifické procesy či obory, přičemž nemusí pokrývat všechny klíčové procesy. Vyznačují se však špičkovou detailní funkcionalitou, ale jejich nevýhodou se může stát obtížnější koordinace procesů nebo nutnost řešení u více IT projektů.
- Lite ERP – jsou odlehčené verze standardního ERP systému orientované na trh malých i středně velkých firem. Výhodou je u nich nízká cena a orientace na

rychlou implementaci. Nevýhodné jsou omezení ve funkcionalitě, dále počtu uživatelů nebo i v možnostech rozšíření (Sodomka, 2010).

1.4.2 Holisticko-procesní klasifikace

Informační systémy firmy je vhodné kategorizovat podle jejich uplatnění v praxi, jak s ohledem na nabídku dodavatelů, tak s ohledem na požadavky na řízení procesů podniku. Zásadní pro klasifikování podnikových informačních systémů je holisticko-procesní hledisko. Podle takové klasifikace tvoří informační systém:

1. ERP jádro, jež je orientované na řízení vnitřních procesů podniku,

2. CRM (Customer Relationship Management) – systém, který obsluhuje procesy směrem ke klientům,

3. SCM (Supply Chain Management) – systém, který řídí dodavatelský řetězec. Jeho součástí bývá APS (Advanced Planning Scheduling), tj. systém, který slouží k pokročilému plánování i rozvrhování výroby,

4. MIS (Management information system) – jedná se o informační systém pro manažery, který sbírá data z CRM a ASP/SCM a ERP systému i z externích zdrojů. Na podkladě těchto informací pomáhá v rozhodovacím procesu managementu daného podniku (Sodomka, 2010).

1.5 Vysvětlení pojmů spojených s informačními systémy

Ohledně výběru informačního systému je možné se setkat s rozličnými typy úloh v aplikační vrstvě:

- MIS (Management Information System) – stanovené pro podporu taktického i operativního řízení. Takovéto úlohy pokrývají procesy, které se týkají oblasti financí, nákupu, prodeje, podporují evidenční a také analytické operace. Informace jsou využitelné jako podklad pro rozhodování.
- EIS (Executive Information System) – stanovené pro manažerské potřeby, využívají přitom dat získaných z úloh typu MIS nebo CIS. Záměrem úloh je také připravit podklad vhodný pro rozhodování.
- DWH (Data Warehouse) – jde o úlohy stylu datový sklad. Jedná se o shromažďování specifických informací z různorodých databází ostatních úloh do jednotného prostředí.
- EDI (Electronic Data Interchange) – zde mají za cíl úlohy obstarat výměnu dat s obchodními partnery nebo dalšími ekonomickými subjekty elektronickou formou.
- OIS (Office Information System) – stanovený k podpoře kancelářských prací, ke zvýšení úrovně pořádku administrativy podniku, urychlení běžné komunikace mezi zaměstnanci firmy nebo s pracovníky externích organizací.
- CAD/CAM (Computer-aided design/Computer-aided manufacturing) – je stanoven pro výrobní úlohy, k optimalizaci řízení výrobních provozů.
- CIS (Customer Information System) – cílem u zákaznických úloh je zajistit základní operace spojené s evidencí spotřeby, dále sledováním pohledávek, zajistit vazby i vstupy do úloh typu MIS (Řepa, 1999).

2 Database design

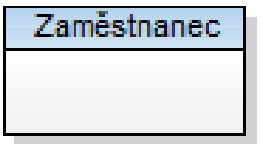
2.1 Teoretické základy datového modelování

Datové modelování je velice obtížná a nezbytná část tvorby programu. Pro její tvorbu je potřeba dodržovat základní kroky a prvky, které jsou popsány níže.

2.1.1 Entity

Entitou je myšleno cokoliv, o čem je potřeba uchovávat informace. Entita představuje soubor objektů, které mají stejné vlastnosti. Zároveň každá entita musí být jednoznačně identifikovatelná, aby nedocházelo k jejich záměně.

Jako příklad pro analýzu entit poslouží rozhovor se zákazníkem: „Zaměstnanec prodá zákazníkovi zboží, které si objednal“. Z této věty jsou patrné čtyři entity: „Zaměstnanec“, „Prodej“, „Zboží“, „Objednávka“.



Obrázek č. 1: Entita Zaměstnanec

Zdroj: Vlastní zpracování

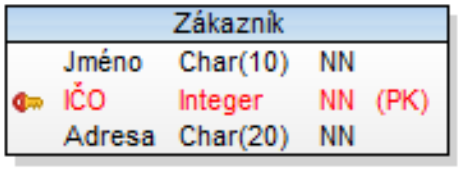
2.1.2 Atributy

Pod pojmem atribut se uchovávají veškeré vlastnosti, které se o dané entitě v databázi potřebují uchovávat. Každý atribut patří vždy a pouze jedné entitě.

Základními pravidly pro tvorbu atributů platí:

Atribut musí být co nejjednodušší. Pro budoucí využití je mnohem snazší propojit jednotlivé vlastnosti než rozpojit dlouhý řetězec vlastností.

Atribut je potřeba navrhnout tak aby byl originální. Například pro člověka je originální rodné číslo. Pro nějaký výrobek je zase originální výrobní číslo.

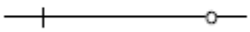


Obrázek č. 2: Entita Zákazník obsahující atributy

Zdroj: Vlastní zpracování

2.1.3 Relace

Pod pojmem relace si můžeme představit propojení mezi dvěma entitami, takové propojení se nazývá binární relace. Pokud je potřeba propojit více entit tak se takové propojení nazývá „vícenásobná relace“. Každé takové propojení musí být jednoznačně identifikovatelné v rámci dané relace. Takovéto jedinečné propojení se nazývá „výskyt relace“. Vztahy s ER diagramu se znázorňují pomocí specifických čar mezi entitami.

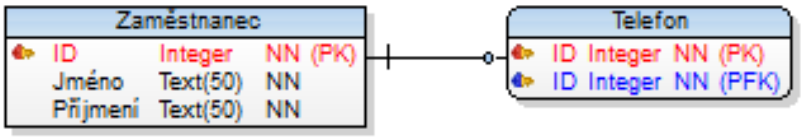


Obrázek č. 3: Základní čára pro relaci

Zdroj: Vlastní zpracování

Relace 1:1

Jediná instance entity A vytváří relaci právě s jedinou instancí entity B. U této relace nám poslouží za příklad entity Zaměstnanec a Telefon. Telefon musí být vždy k zaměstnanci přidělen, ale ne každý zaměstnanec má telefon k dispozici.

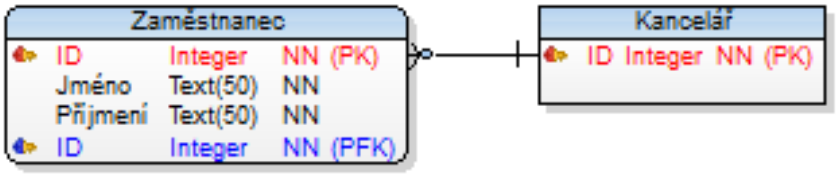


Obrázek č. 4: Relace 1:1 entity Zaměstnanec a entity Telefon

Zdroj: Vlastní zpracování

Relace 1:N

Tato relace je nejpoužívanější při tvorbě rozsáhlých databází. Jako příklad nám poslouží entita Zaměstnanec a Kancelář. Zaměstnanci jsou rozmístěni do kanceláří, ve kterých může být více zaměstnanců. Každý zaměstnanec ale musí mít nějakou kancelář. Na druhou stranu mohou být některé kanceláře prázdné.



Obrázek č. 5: Relace 1:N entity Zaměstnanec a entity Kancelář

Zdroj: Vlastní zpracování

Relace N:N

Instance entity A vstupuje do vztahu s vícero výskyty entity B. Zároveň může jeden výskyt entity B vstoupit do vztahu s více objekty entity A. Zde nám jako příklad poslouží opět entita Zaměstnanec a nová entita Projekt. Zaměstnanec se obvykle účastní více projektů. Při velkém projektu je ale potřeba více zaměstnanců.

2.1.4 Doména

Doména je souhrn všech možných hodnot, které jsou přípustné pro daný atribut. Doména je tudíž logický pojem a nejčastěji se pro její pojem využívá hodnota NULL. Muže se ale také využít výčet hodnot které jsou přípustné. Například při založení si průkazu na MHD je důležitý věk. Pokud je věk žadatele v rozmezí 15 až 18 let tak bude mít studentskou slevu.

2.1.5 Klíč

Každý záznam, který je uvedený v tabulce musí být jedinečný. To znamená, že v tabulce musíme být schopni určit sloupec anebo kombinaci více sloupců který tuto jedinečnou kombinaci zajišťuje.

Základní klíče dělíme na „Super klíč“, „Kandidátní klíč“, „Primární klíč“ a „Cizí klíč“.

Super klíč je definován jako množina atributů proměnné relace, pro kterou platí, že ve všech vztazích přiřazených k této proměnné neexistují dva shodné řádky.

Kandidátní klíč označuje sloupec nebo jejich kombinaci, ve které mají všechny řádky unikátní hodnoty.

Primární klíč je nezávislou entitou, která jednoznačně určuje daný výskyt objektu v databázi. Nikdy nesmí obsahovat NULL a každá tabulka musí mít právě jeden primární klíč.

Cizí klíč určuje vztah mezi dvěma tabulkami, právě tak, že hodnota uvedená v první tabulce musí existovat i v druhé.

2.1.6 Data

Data jsou množinou nezpracovaných neboli surových faktů, která mají určitou důležitost pro jedince, společnost a organizaci. Tato důležitost není nijak významná, jelikož se v množství neutříděných dat špatně orientuje. Tato data se vyskytují v nejrůznějších podobách a v dnešní informační době v obrovském množství. Pro získání dat je potřeba provést pozorování, měření a následný zápis. Jednou z nejčastěji používaných metod pro získání dat je vytvoření dotazníku. Díky této technice získáme tzv. „surová data“ které díky interpretaci převedeme na informace.

2.1.7 Informace

Informace jsou uceleným souborem dat. Data, která byla interpretována člověkem, prošla zpracováním za pomoci počítače nebo dostala nějakou strukturu se stávají informací. Dalším způsobem transformace dat je využití grafických ukazatelů, nebo statistických výpočtů.

Tyto informace mají díky zpracování mnohem větší význam než samotná data. Význam tkví hlavně v ucelení dat do jediného výsledku. Tento výsledek je možné vyvrátit opakovaným sběrem, analýzou a stanovením nového výsledku. Nicméně výsledek je platný pouze do doby, než se nám změní vstupní data.

2.2 Tvorba databáze

Databáze tvoří většinou základní součást informačního systému, a to především u obchodně orientovaných systémů.

Do plánování databáze spadají činnosti, jež umožňují stádia vývoje databázového systému, jeho životního cyklu, jež mají být zrealizovány efektivně i účinně, jak to jen lze. Je potřeba integrovat do celkové strategie informačního systému podniku.

Prvním krokem u plánování databází je definovat poslání a cíl pro daný databázový systém, účel databázového systému, dále podporované úkoly takového systému a prostředky.

Fáze definování systémů, rozsahu a hranice databázové aplikace obsahuje vyřešení těchto kroků:

- propojit s ostatními informačními systémy podniku,
- co má plánovaný systém konat nyní i v budoucnosti,
- kdo bude současnými uživateli a kdo také v budoucnu.

Je třeba znát i názory uživatelů, tj., co je očekáváno a žádáno od databázového systému z hlediska jednotlivých pracovních rolí nebo aplikačních oblastí podniku.

Důležité je nezapomenout také požadavky na sběr a analýzu informací, jež mají být podávány v databázi jako dokončené. Výsledkem může být např. popis dat užívaných nebo generovaných a jiné.

Fáze, kterou lze nazvat konstrukcí databáze je dělena do těchto tří kroků:

- koncepční návrh u databáze
- logický návrh databáze
- fyzický návrh databáze.

2.2.1 Konceptuální úroveň

První úroveň je tak zvaná konceptuální úroveň. V této úrovni se zabýváme výpisem a popisem veškerých informací, které budeme potřebovat pro tvorbu a chod celého databázového systému. V této úrovni se záměrně věnujeme pouze informacím, které se v následujících úrovních budou nadále využívat a zároveň neřešíme, jakým způsobem tyto informace použijeme.

Nejběžnějším zápisem informací v první vrstvě je pomocí ER diagramu.

2.2.2 Logický model

Ve fázi logické návrhu je řeč o modelu údajů, jež mají být užity se založením na konkrétním modelu dat. Při převodu konceptuálního modelu na relační schéma se musí dodržet následující pravidla. Každá entita je samostatnou tabulkou. Identifikátorem entity se stává primární klíč. (Šimůnek, 2001)

Entity - Relationship Diagram

ER Diagramy graficky popisují vztahy mezi entity. Tyto diagramy zobrazují myšlenkový databázový model. Diagramy jsou zcela nezávislé na znalosti budoucí fyzické implementaci databáze. ER Diagramy jsou jedním z nejdůležitějších nástrojů pro tvorbu návrhu databáze.

Každý symbol, který se může vyskytnout v ER Diagramu má přesně definovanou funkci. Jako čtverec nebo kolečko se obvykle vyskytuje entita. Pod entitou se může skrývat člověk, bydliště, kancelář. Propojení těchto entit má na starosti kosočtverec, který udává vztah mezi entitami. Elipsa je funkce, která je propojena s entitou a udává jakou operaci má entita udělat.

2.2.3 Fyzický model

Ve fyzické fázi návrhu je vytvořen popis implementace databáze na sekundární úložiště (tedy základní vztahy, bezpečnost indexy a jiné jsou definovány pomocí jazyka SQL).

Ve fázi návrhu aplikace, designu uživatelského rozhraní a u aplikačních programů, jež se užívají a zpracovávají u databáze, je třeba vše potřebné definovat.

Účelem prototypu je, aby bylo umožněno uživatelům užívat prototyp, stanovit vlastnosti a chování systému za pomocí počítače. K dispozici jsou přitom horizontální a také vertikální prototypy. Horizontální Prototyp má řadu funkcí (například uživatelských rozhraní). Vertikální prototyp má velmi málo funkcí.

Nežli se nový systém spustí do provozu jako takového, je třeba důkladně testovat pro odhalení nežádoucích chyb, přičemž cílem není dokazovat, že software funguje dobře.

Podstatná je také posléze provozní údržba, kdy dochází k pravidelnému monitorování a udržování systému dané databáze, tedy je sledován výkon systému. V případě, že výkon systému poklesne pod přijatelnou úroveň, nastává fáze ladění či reorganizace databáze, která je potřebná (Auer, 2006).

Údržba a modernizace databázového systému je nezbytná také, když vzniknou nové požadavky. (Connolly, Begg, 2005). Nové verze systému je nejlepší plánovat dopředu, než vyjde novější verze například operačního systému, který starou verzi nespustí.

2.3 Normalizace databáze

Normalizace databáze je soubor postupů, kdy jsou data v tabulce přeorganizována, aby se co nejvíce využilo výhod relační databáze. Díky normalizaci se zajistí lepší ukládání dat, jejich třídění a prohledávání celé tabulky. Jedna z hlavních úvah, která je spjata s normalizací zní: „Kam až normalizovat“. Tato úvaha je zcela subjektivní a závisí na rozhodnutí každého kdo tvoří databázi.

2.3.1 0NF

V nulové normální formě jsou všechny atributy potřebné v databázi jsou bez jakéhokoliv třízení jednoznačně pojmenovány, avšak neustále se opakují. Například „Zákazník jméno“, „Zákazník adresa“, „Zákazník IČO“, „Zaměstnanec jméno“ a „Zaměstnanec pozice“.

Tito atributy jsou sice jednoznačné ale nejsou propojené do dvou entit.

2.3.2 1NF

Díky první normální formě se nám atributy z příkladu víše rozdělí do dvou entit a to, entita „Zákazník“ a entita „Zaměstnanec“.

Pod entitu „Zákazník“ se uloží atributy „jméno“, „adresa“ a „IČO“.

Pod entitu „Zaměstnanec“ se uloží pouze: „jméno“ a „pozice“.

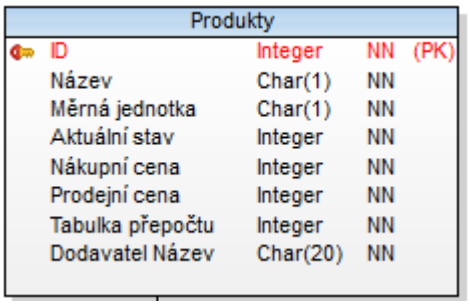
Díky rozdělení do 1NF se nám vytvořily dvě tabulky a výrazně se nám zjednodušilo vyhledávání a ukládání dat.

2.3.3 2NF

Entita naprosto vyhovuje druhé normální formě, pokud se nachází v první normální formě a zároveň jsou všechny její neklíčové atributy funkčně závislé na kandidátním klíči. To znamená, že pokud tabulka, která obsahuje duplicitní položky a vytváří mezi sebou částečnou závislost je potřeba rozdělit do nových tabulek kde bude údaj uložen pouze jednou (Malsakowski, 2001).

Jako příklad poslouží entita „Faktura“ s atributy „FakturaČíslo“, „DatumSplatnosti“, „ZbožíID“ a „Množství“. V takovéto entitě je kandidátním klíčem kombinace atributů „FakturaČíslo“ a „ZbožíID“. Atribut „DatumSplatnosti“ je ale závislý pouze na atributu „FakturaČíslo“ a ne na celém kandidátním klíči, a proto taková entita není ve druhé normální formě.

Pro vytvoření 2NF přesuneme atributy „FakturaČíslo“ a „DatumSplatnosti“ do samostatné entity „Hlavičkafaktury“, a atributy „FakturaČíslo“, „ZbožíID“ a „Množství“ do druhé entity „Řádkyfaktury“. Obě relace pak budou ve druhé normální formě. Atribut „FakturaČíslo“ v entitě „Řádkyfaktury“ bude opět cizím klíčem z entity „Hlavičkafaktury“.

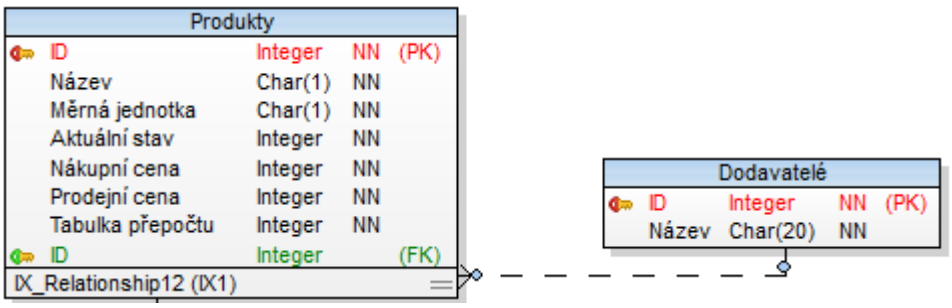


Obrázek č. 6: Tabulka v 2NF

Zdroj: Vlastní zpracování

2.3.4 3NF

Pravidla pro třetí normální formu říká, že je potřeba důsledné odstranění a oddělení veškerých dat, které nejsou v přímé relaci s primárním klíčem dané entity. Hodnota každého pole musí být závislá na hodnotě klíčové položky, která naprosto přesně identifikuje všechna pole záznamu. Nevhodné pole pro třetí normální formu odebereme z entity a vytvoříme mu novou tabulku. Hodnotu z této tabulky propojíme pomocí cizího klíče do původní tabulky.



Obrázek č. 7: Tabulka v 3NF

Zdroj: Vlastní zpracování

Pokud tabulka Zákazník obsahuje zároveň pole Dodavatel_Název tak je potřeba pro pole Dodavatel_Název vytvořit novou tabulku s názvem dodavatel která bude obsahovat pole Název. Po této úpravě tabulka spadá do třetí normální formy.

2.4 Databázový systém

Za databázový systém je považováno sloučení dat a nástrojů, jejichž pomocí vytváříme, aktualizujeme, vyhledáváme a rušíme získaná data.

Veškeré databázové systémy obsahují nástroje pro definici struktury dat, zajištění fyzické a logické nezávislosti dat. Může také obsahovat nástroje zálohování dat a podporu práce více uživatelů.

2.5 Datové typy

Pro snadnější pochopení nám poslouží tabulka, která obsahuje „Jméno“, „Příjmení“, „Datum narození“ a „Výše platu v Kč“. Základními prvky tabulky jsou řádek, sloupec a hodnota. Z této tabulky je na první pohled patrné že obsahuje tři datové typy.

Znak nebo text je základní datový typ. Sloupce tohoto typu mohou v tabulce obsahovat libovolné znaky z abecedy, znaky jako jsou čárky, pomlčky, středníky atd. anebo číslice.

Číselný datový typ obsahuje pouze hodnoty přednastaveného rozsahu. Základní rozlišení patří, jestli se uvažuje o desetinném či celém čísle. Pro obor celých čísel rozlišujeme rozsah – např. od 0 do 255, od -32768 do 32767 apod. U desetinných čísel určíme pouze počet číslic za desetinnou čárkou. (Šimůnek, 2001)

Datový typ datum umožňuje uchování data v libovolném formátu. K nejběžnější uchování datumu slouží formát (den-měsíc-rok;DD/MM/RRRR). Ale i tento nejběžnější formát lze nastavit podle potřeby.

Příjmení	Jméno	Datum narození	Výše platu v Kč
Bubník	Filip	15.03.1989	17 000,00
Kolář	Lukáš	19.08.1975	25 000,00
Veselá	Martina	02.12.1993	14 000,00

Obrázek č. 8: Tabulka datové typy

Zdroj: Vlastní zpracování

Ano/Ne je datový typ, který slouží pro uchovávání pouze dvou hodnot. Ano (pravda) a Ne (nepravda). V praxi nám tato logická hodnota může uchovávat informaci ohledně GDPR. Pokud získáme souhlas se zpracováním osobních údajů tak bude hodnota v proměnné „pravda“ a pokud ne tak se zadá „nepravda“.

2.6 SQL (Structured Query Language)

SQL je relační databázový systém, který je založen na relačním modelu dat a na relační algebře (Šimůnek, 2001). První relační databáze vznikla v 70. letech 20. století ve firmě IBM. Tato databáze měla zkratku SEQUEL a jazyk SQL z ní vychází.

Operace relační algebry

Pro veškeré operace mezi dvěma a více relacemi je potřeba, aby právě tyto dvě nebo více relací měly naprosto stejnou strukturu. Hlavními operacemi jsou projekce, restrikce a spojení tabulek.

Příkazy

Příkaz Select slouží pro výběr dat z databáze. Umožňuje výběr podmnožin a řídit data.

Insert slouží pouze ke vkládání nových dat do databáze.

Příkaz Update mění data v databázi.

Kombinací příkazů Insert a Update vzniká příkaz Merge. Pokud data neexistují tak je vloží a pokud existují tak je upraví.

Příkaz Delete odstraní určená data z databáze.

2.7 DLL (Dynamic-link library)

DLL je knihovna která obsahuje data a kód, který je možné využívat více programy zároveň. Použitím DLL lze program rozdělit modul na samotné komponenty. Při aktualizacích lze aktualizovat jednotlivé moduly, aniž by byly ovlivněny jiné části programu.

Knihovny DLL nelze spouštět samostatně a pro jejich spuštění je potřeba vytvořit exe soubor. Tyto knihovny se spouští v rámci volaného procesu nebo aktuálně používaného programu kde jsou sdílena přístupová práva a paměťový prostor.

3 Tvorba skladového informačního systému

Samotná tvorba informačního systému neznamená pouze programování aplikace. Před samotným zahájením programování je potřeba provést analýzu potřeb a funkcí které má budoucí program splňovat. Následuje tvorba logického modelu, kde je zapotřebí popsat veškeré funkce, které bude potřeba v další fázi naprogramovat. Pří programování aplikace je potřeba dodržovat předchozí výstupy z analýzy informačního systému.

Jako poslední krok je na řadě testování a tvorba nových verzí které zajistí správně fungování programu při změně zákonů či programovacích jazyků.

3.1 Definice požadavků

Ve středu zájmu při implementaci nového informačního systému je vlastní realizační fáze, která obsahuje zavádění i nastavení aplikace, další fáze, které jsou však často podceňovány. Pokud mají podniky vymezit, co od budoucího systému očekávají, sklouzávají často jen k obecným formulacím jako např. „potřebujeme účetnictví, jsme klasická firma“. Takovýto přístup však vede jen k tomu, že při následné fázi výběru pomyslným sítem projde obšírný počet systémů i aplikací, jež mají velký rozsah funkcionality. Z toho se posléze ale také odvíjí i cena takovýchto systémů a také fáze výběru se stává záležitostí pro několik výběrových kol, řady týdnů či dokonce měsíců. Po skončení výběru by mohl navíc zůstat u řady manažerů nebo klíčových pracovníků váhavý pocit, zda skutečně byl vybraný systém tou správnou volbou.

Opět se tedy potvrzuje, že úvodní fáze celého procesu je v tomto pojetí jednou z nejpodstatnějších a podle její kvality se odvíjí i výsledná volba produktu. Managementu podniku dává tato fáze také prostor k uspořádání vlastních představ i očekávání, co má takový informační systém pro potřeby organizace plnit.

Neméně podstatným prvním krokem je stanovit odpovědné jedince za tvorbu zadání (tedy i vedoucího projektu) a definovat samotný pracovní tým. Vedoucí projektu přitom nemusí být nezbytně osoba jen z managementu organizace, avšak pro správný průběh je nezbytné vytvořit funkční „komunikační kanál pro management“. Složení samotného týmu se odvíjí od celkového rozsahu procesů a aktivit, jež má budoucí systém pokrýt. Pro každou oblast/proces má firma vymezit klíčového pracovníka (garanta), jež je pak odpovědnou osobou za definování kritérií a který podle potřeby obstará také komunikaci dovnitř

firmy. Praxe ukázala, že pro účinné fungování je nezbytné udržet pracovní tým v rozumném rozsahu cca od 6 až do 10 osob.

Obvyklá úskalí při tvorbě zadání:

- vytváření zadání je třeba věnovat dostatečně dlouhý čas a pozornost,
- do tvorby zadání se musí zapojit i management podniku, který zná požadavky na systém,
- systém nebude přitom existovat samostatně „v prostoru“, musí být tedy včleněn do prostředí firmy,
- i v případě obměňování informačního systému je nezbytností zhodnotit požadavky na danou funkcionalitu, tedy argument jako například „teď to v IS máme“ se jeví nedostatečným,
- korekce množství požadavků na systém by měla být v rozumné míře, jelikož podle zkušeností je převážná část systémů využívána pouze z 30-50 % svých možností, v budoucnu nevyužitá funkcionalita přitom může stát firmu nemalé finance,
- součástí zadání má být stanovení i přínosů při správném využití plánovaného informačního systému.

K dalším oblastem, na které se předem obvykle v úvodních diskusí zapomíná či jsou řešeny pouze z části, patří vlastní oblast u IT. U převážné části případů je takováto oblast řešena určením primárních technických podmínek pro provoz daného systému – serverová část, typ a verze u databázového serveru, HW požadavky, operační systém na pracovních stanicích. Ohledně oblastí IT je třeba vyřešit i celkové začlenění informačního systému do infrastruktury podniku. Je nutné zvážit, které stávající systémy budou s daným novým nástrojem komunikovat a u kterých bude třeba vytvořit nebo modifikovat datová rozhraní. Jako součást zadání musí být také informace o užívaných standardních SW nástrojích, jako je například typ i verze kancelářského balíku, který produkt podnik využívá pro emailovou komunikaci. U celé řady projektů opomenutí takovýchto zdánlivě samozřejmých nebo jednoduchých věcí posléze generuje vyvolané investice (například vynucená obměna u kancelářského balíku na aktuálnější verzi), (Rek, 2011).

3.2 Analýza skladového informačního systému

Při analýze požadavků na nový skladový informační systém se musí zohlednit velikost firmy, množství skladů, množství položek, jedna či více měn, počet zákazníků a dodavatelů. Zároveň se nesmí zapomenout na nezbytně nutné požadavky ani na požadavky které v budoucnu mohou rozšířit funkcionalitu informačního systému.

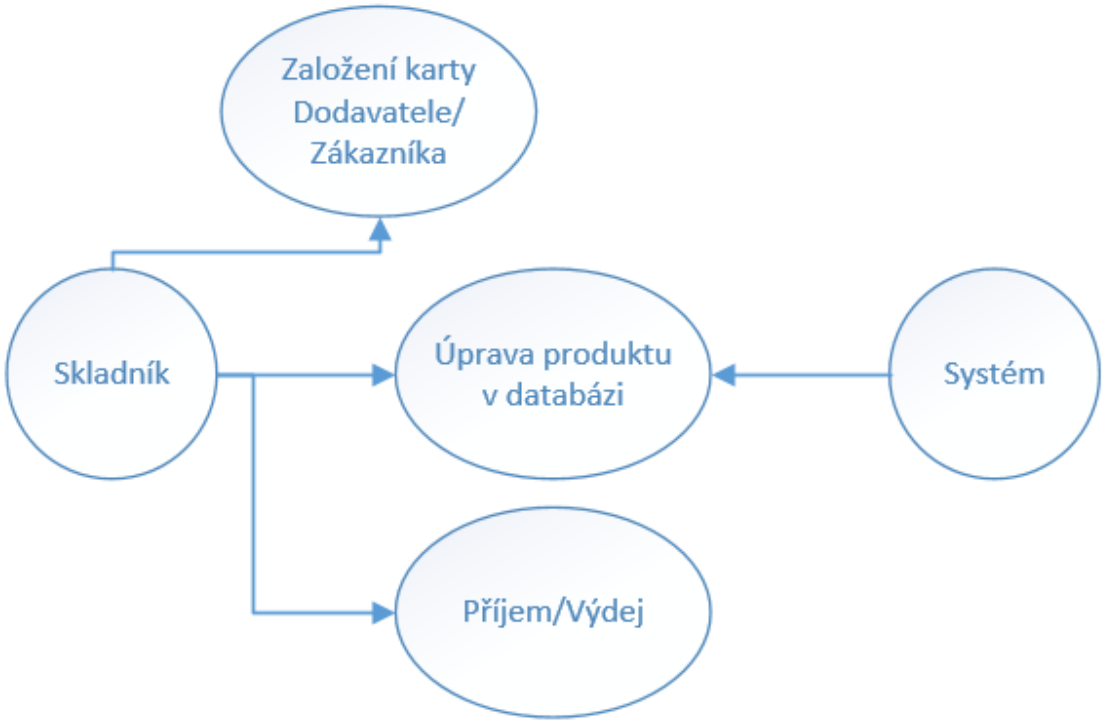
Hlavní požadavek je, aby uživatel mohl plně využívat skladový program pro evidenci produktů. Na tuto evidenci produktů se vztahuje právo vytvářet nové karty jednotlivých produktů, měnit ceny, množství. Další potřebná funkce je evidence dodavatelů. Pro tuto evidenci bude vytvořena entita, do které bude mít uživatel přístup přes formulář. Stejná funkce platí také pro evidenci zákazníků.

3.3 Konceptuální model

Pro tvorbu konceptuálního modelu budoucí aplikace jsem zvolil program Visio Profesional 2016 od firmy Microsoft. Tento program umožňuje tvorbu celistvého návrhu aplikace.

Konceptuální model je navržen tak, aby měl skladník veškerá oprávnění pro kontrolu a tvorbu celého programu. Skladník má možnost vytvořit, upravit a smazat kartu Zákazníka nebo Dodavatele. Taktéž má plné oprávnění pro tvorbu nové Příjemky či Výdejky zboží na nebo ze skladu.

Systém slouží k doplňování dat, které si načte z Příjemky nebo Výdejky do tabulky Produkty.

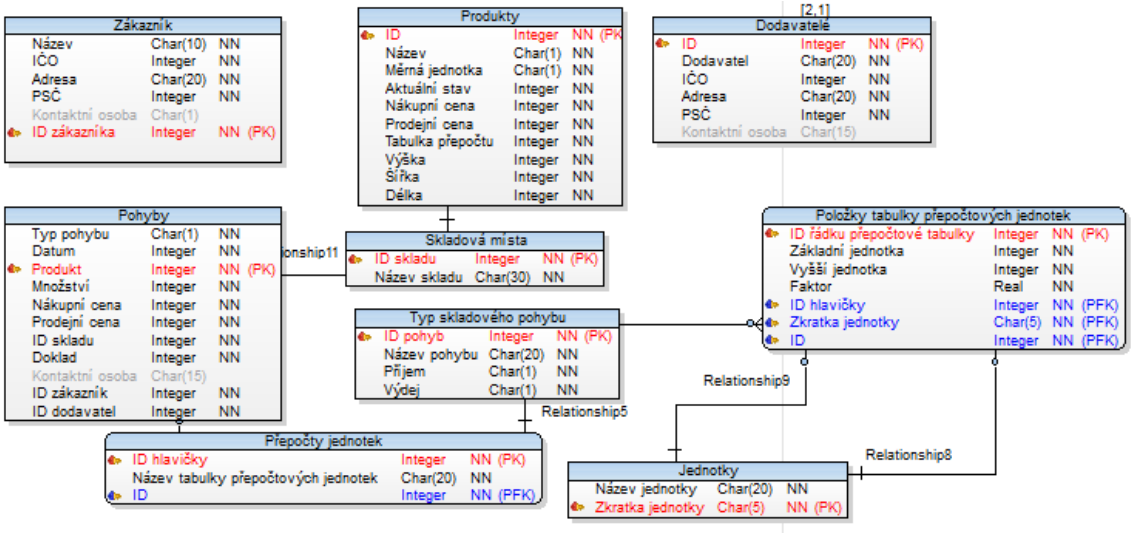


Obrázek č. 9: Konceptuální model

Zdroj: Vlastní zpracování

3.4 Logický model

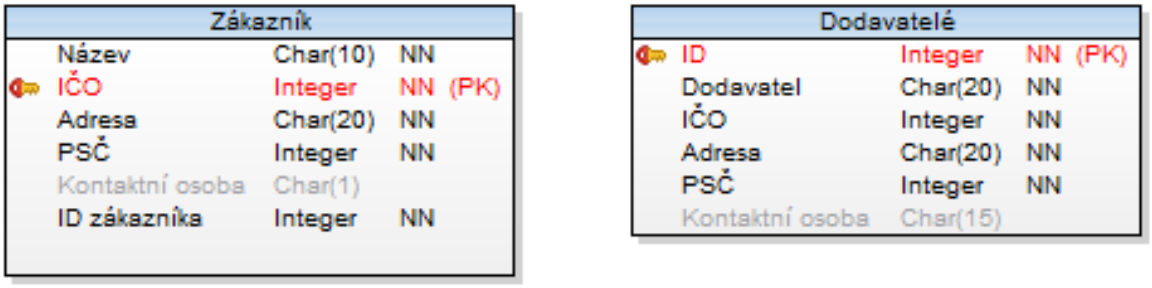
K tvorbě ER diagramu byl použit program Toad Data Modeler od firmy Quest Software. Relace, které jsou zobrazen níže jsou základem celé aplikace.



Obrázek č. 10: Relace

Zdroj: Vlastní zpracování

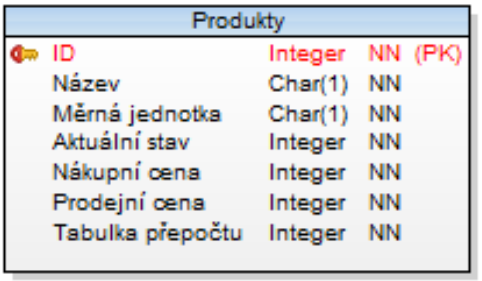
Dvě základní entity se nazývají „Zákazník“ a „Dodavatel“. Každá z nich obsahuje specifické ID, které slouží jako primární klíč a pod tímto ID bude daný zákazník nebo dodavatel evidován v databázi. Dalšími atributy jsou název, adresa a PSČ. Tyto údaje jsou velice důležité pro případnou fakturaci. Poslední údaj je IČO, které je u dodavatele povinné ale u zákazníka již povinné není, a to z důvodu prodeje zboží jak fyzickým, tak i právnickým osobám.



Obrázek č. 11: Entity Zákazník a Dodavatel

Zdroj: Vlastní zpracování

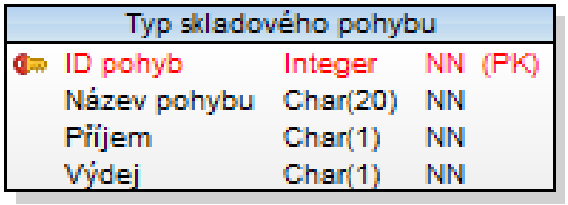
Entita „Produkty“ obsahuje atributy, které jsou potřeba pro přesné určení zboží uskladněného na skladě. Každé zboží má specifické ID, které opět slouží jako primární klíč. Při pohledu na řádek produktu skladník okamžitě zjistí název, aktuální stav na skladě, měrnou jednotku přiřazenou danému produktu, rozměry, nákupní a prodejní cenu.



Obrázek č. 12: Entita Produkty

Zdroj: Vlastní zpracování

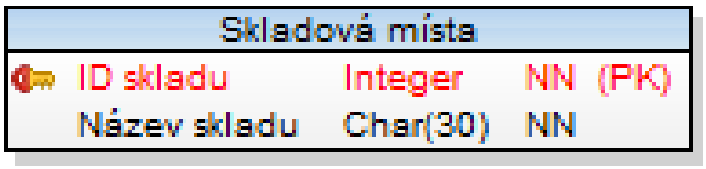
Tabulka „Typ skladového pohybu“ obsahuje primární klíč „ID pohyb“, atribut „Název pohybu“ a bolean proměnnou hodnotu „Příjem“ nebo „Výdej“. Tyto atributy poskytují veškeré informace, které jsou zapotřebí pro rozpoznání pohybu na skladu.



Obrázek č. 13: Entita Typ skladového pohybu

Zdroj: Vlastní zpracování

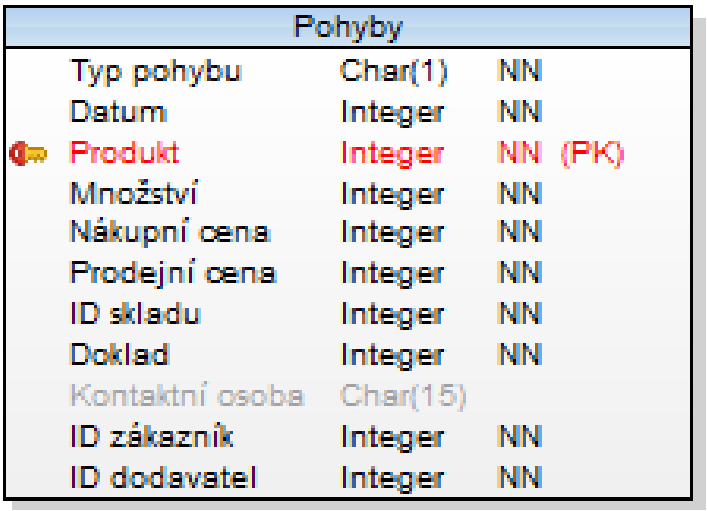
Entita „Skladová místa“ nám udává „ID skladu“ a jeho název. Díky využití informací z této tabulky budeme naprosto přesně vědět kde je dané zboží uložené. Tabulka skladová místa je následně využita pomocí cizího klíče v tabulce produktů.



Obrázek č. 14: Entita Skladová místa

Zdroj: Vlastní zpracování

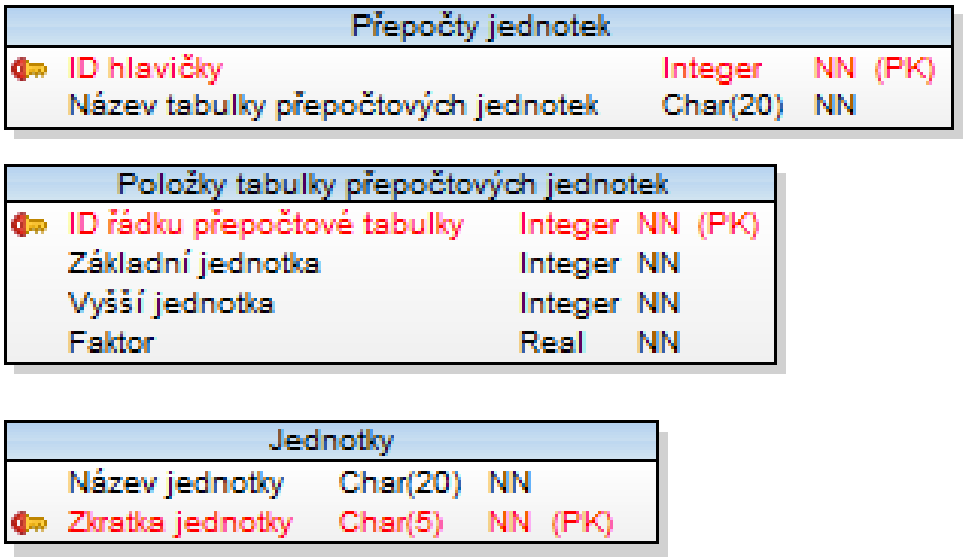
Velmi důležitá tabulka „Pohyby“ obsahuje informace ze všech ostatních entit. Každý pohyb, ať už příjem nebo výdej má generováno svoje jedinečné ID. Toto ID se určuje dle typu skladového pohybu. Pokud se bude jednat o přijetí zboží lze před číselnou hodnotou zadat například symbol „P“, při výdeji zboží lze před číselnou hodnotu nastavit písmeno „V“ dle požadavku zákazníka. V souvislosti s výběrem pohybu se zpřístupní k vyplnění „ID dodavatel“ nebo „ID zákazník“. Jako primární klíč zde slouží atribut „Pohyb ID“. V neposlední řadě je v této tabulce uložena nákupní a prodejní cena, množství, ID skladu, kde bude zboží uloženo nebo vydáno a související ID dokladu.



Obrázek č. 15: Entita Pohyby

Zdroj: Vlastní zpracování

Poslední tabulky „Jednotky“, „Přepočty jednotek“ a „Položky tabulky přepočtových jednotek“ slouží k přepočtu měrných jednotek. Dle požadavků zákazníka lze přednastavit základní měrné jednotky a jejich ekvivalenty. Jako příklad nám poslouží jeden kus zboží, vyšší jednotka se stanoví karton, který pojme 6 kusů zboží. Pokud, bychom potřebovali vyšší jednotku tak zákazník použije přednastavenou paletu, která pojme například 24 kartonů.



Obrázek č. 16: Entity pro přepočet jednotek

Zdroj: Vlastní zpracování

3.5 Tvorba aplikace

Pro samotnou tvorbu aplikace jsem si zvolil program Access od firmy Microsoft. Tento program slouží k tvorbě nejen databázových aplikací. Díky kombinaci několika významných jazyků se z Accessu stává velice kvalitní program. Access využívá syntaxi SQL jazyka, Visual Basic for Applications a velice užitečná jsou makra pro usnadnění pohybu mezi jednotlivými tabulkami, dotazy a formuláři.

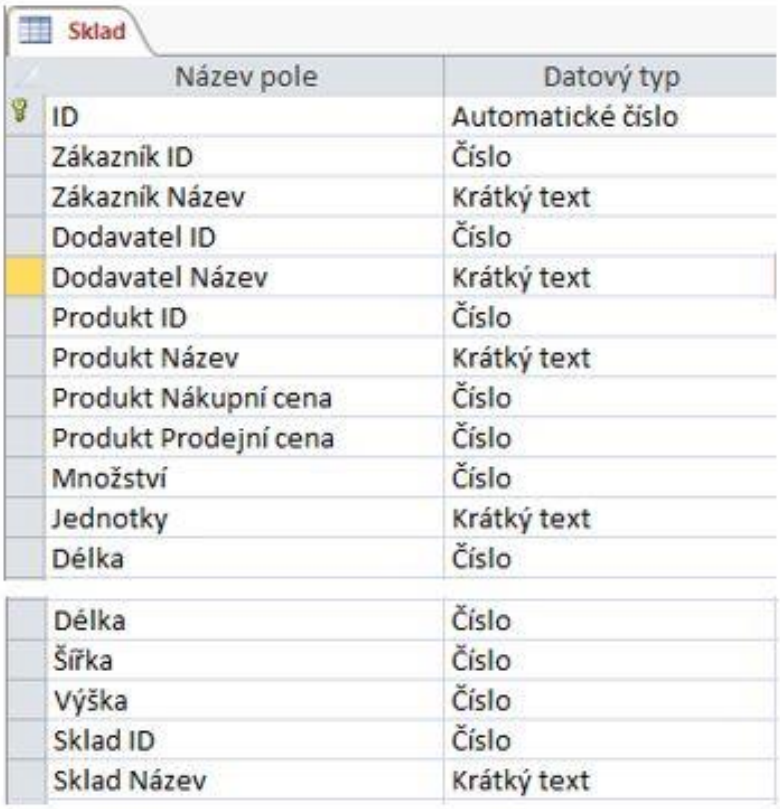
Převod SQL do programu Access

Z logického modelu, který jsem v předchozí kapitole popsal a vytvořil jsem vygeneroval SQL kód který umožňuje budoucí použití v aplikaci Access. Tento SQL kód se snadno nahraje a během několika málo okamžiků tvoříme budoucí aplikaci pro skladovou evidenci.

3.5.1 Aplikace před normalizací

Tvorba aplikace pro 0NF probíhala vytvořením jediné entity pod názvem „Sklad“. Tato entita obsahovala veškeré atributy, které jsou potřeba pro chod programu. Každá skladová jednotka obsahuje informace, které jsou potřeba vyplnit u každé položky a některé jsou vázány na druh pohybu položky ve skladu. V případě příjmu zboží na sklad se nevyplňuje „Zákazník ID“ ani „Zákazník Název“. Všechny ostatní řádky jsou povinné a skladník je před uložením záznamu musí vyplnit.

Tato verze databáze působí tak, že je vše pohromadě ale pro její budoucí úpravu je naprosto nevyhovující. Také vyhledávání je velice obtížné z důvodu nekonzistence dat a možného opakování záznamů.



Sklad

Název pole	Datový typ
ID	Automatické číslo
Zákazník ID	Číslo
Zákazník Název	Krátký text
Dodavatel ID	Číslo
Dodavatel Název	Krátký text
Produkt ID	Číslo
Produkt Název	Krátký text
Produkt Nákupní cena	Číslo
Produkt Prodejní cena	Číslo
Množství	Číslo
Jednotky	Krátký text
Délka	Číslo
Délka	Číslo
Šířka	Číslo
Výška	Číslo
Sklad ID	Číslo
Sklad Název	Krátký text

Obrázek č. 17: Databáze v 0NF a datové typy

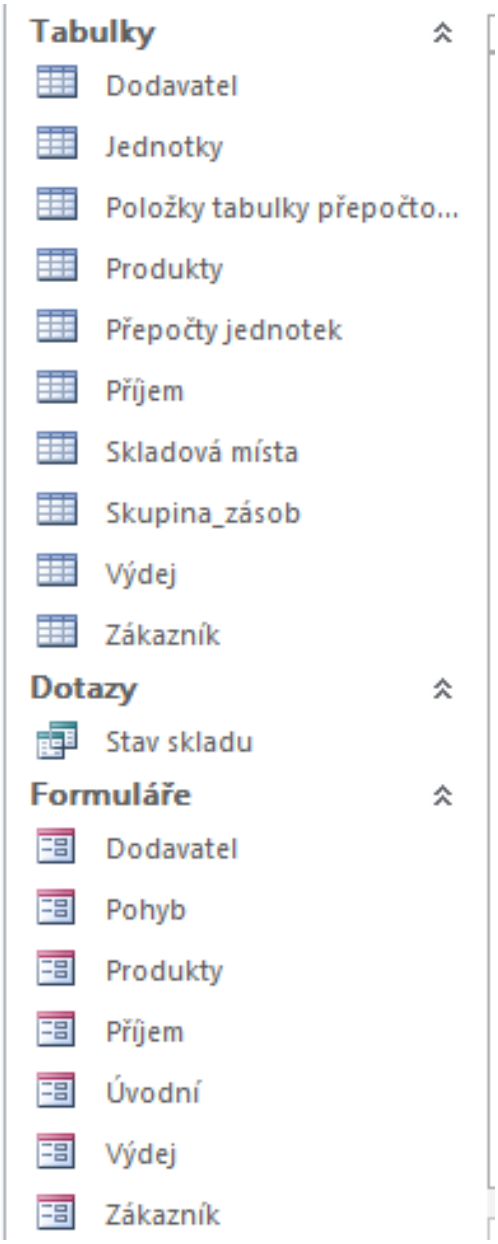
Zdroj: Vlastní zpracování

3.5.2 Normalizace databáze

Na začátku normalizace je potřeba si říci do jaké normální formy chceme databázi upravit. Já jsem si vybral třetí normální formu začal jsem tvořit nové entity. Entitu sklad, která je v nulové normální formě rozdělíme celkem do 8 nových tabulek. Nové tabulky jsou pojmenovány „Dodavatel“, „Zákazník“, „Příjem“, „Výdej“, „Jednotky“, „Produkty“, „Skladová místa“ a „Skupina_zásob“. Toto rozdělení již splňuje rozdělení entit dle třetí normální formy.

Veškeré atributy, které jsou v jednotlivých entitách jsou k dispozici k nahlédnutí v obrázku č. 18.

Tato verze databáze byla vytvořena pro konečnou aplikaci. Skladová databáze díky rozdělení do nově vytvořených tabulek začala splňovat normy pro třetí normální databázi.

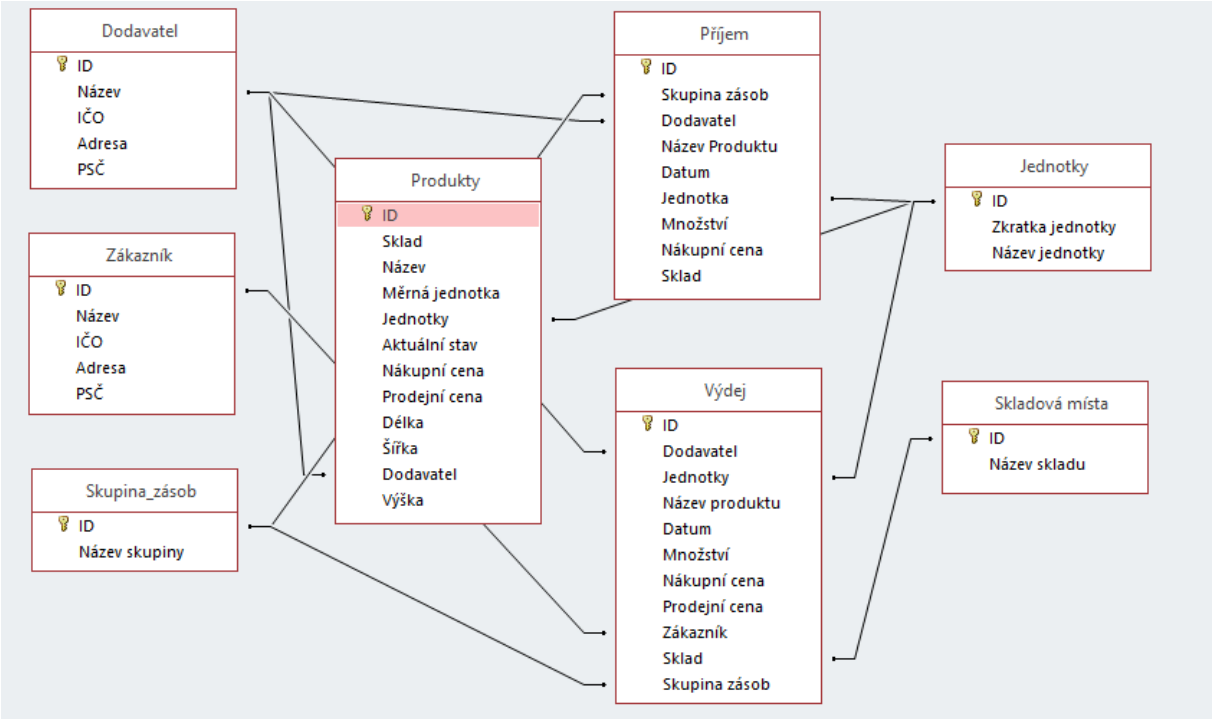


Obrázek č. 18: Tabulky, dotazy a formuláře

Zdroj: Vlastní zpracování

Obrázek č. 18 zobrazuje veškeré tabulky a formuláře které jsou k dispozici. Tabulky byly převedeny z logického modelu díky exportu z programu Toad Data Modeler do SQL kódu, který je možné importovat do programu Access.

Formuláře byly vytvořeny z jednotlivých tabulek. Všechny tabulky ale nepotřebují vlastní formulář, přes který by uživatel mohl měnit obsah záměrně nepřístupných tabulek.



Obrázek č. 19: Relace 3NF

Zdroj: Vlastní zpracování

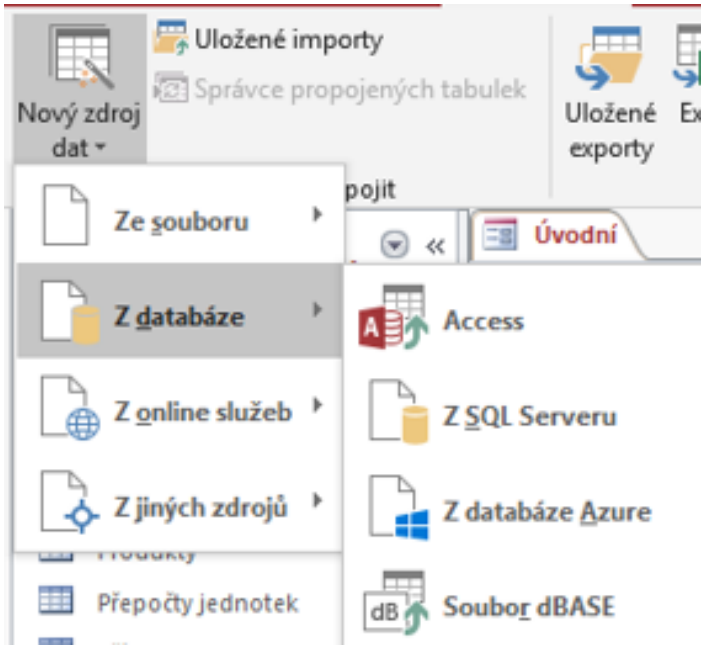
Atributy byly záměrně voleny tak aby byly pokryty veškeré potřeba na nově tvořený informační systém. Nejobsáhlejší entita „Produkty“ obsahuje atributy, které obsahují veškeré informace o ceně, dodavateli, uložení na skladu o každém jednotlivém produktu.

Pomocí relací mezi tabulkami je možné pomocí cizího klíče zpřístupnit data mezi nimi. Tyto relace jsou naprosto nezbytné pro fungování programu.

Pokud by uživatel potřeboval více atributů tak si v tabulce může vytvořit nový sloupeček kterému nadefinuje datový typ a může jej propojit s jinou entitou.

3.6 Tabulky

Tabulky, které jsou potřeba pro nově tvořenou aplikaci byly vytvořeny automaticky za pomoci importu SQL souboru. Program Access nabízí import databáze ve více formátech a nejen SQL.



Obrázek č. 20: Import SQL

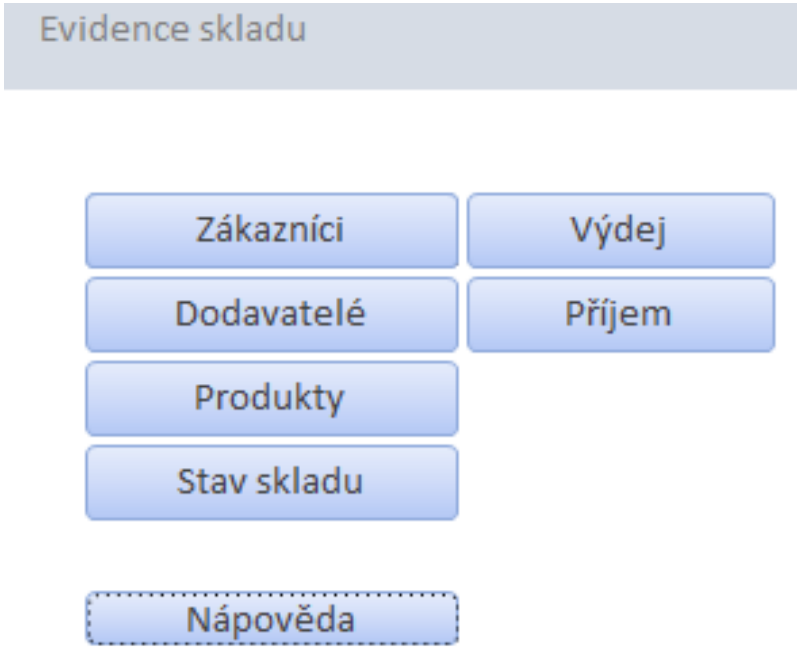
Zdroj: Vlastní zpracování

Následná úprava tabulek spočívala ve správném výběru datového typu každého atributu v každé entitě. Nejčastějšími datovými typy byly hodnoty „Číslice“, anebo „Krátký text“.

Když byly nadefinovány datové typy tak jsem naplnil základními parametry tabulku „Jednotek“ předem stanovenými hodnotami „Kus“, „Krabice“ a „Paleta“. První data jsem ručně vyplnil i do tabulek „Zákazník“ a „Dodavatel“ pro budoucí tvorbu formulářů ve kterých tyto data budou načteny.

3.7 Formuláře

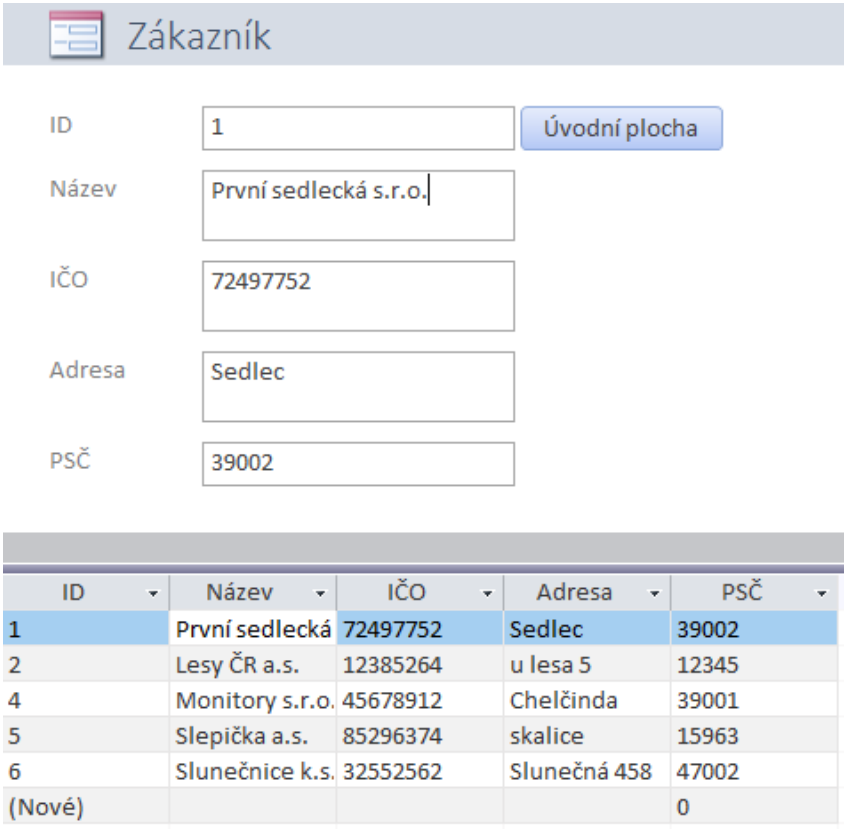
Aplikace obsahuje základní úvodní menu, které je při každém spuštění automaticky zobrazeno. Toto menu je vytvořeno jako formulář a obsahuje 6 hlavních odkazů které uživatele přesměrují do jiných částí aplikace. Dodatečné tlačítko „Nápověda“ obsahuje informace, jak správně využívat tento program. Design aplikace není nijak zvlášť řešen, jelikož jsem se zaměřoval na funkčnost.



Obrázek č. 21: Úvodní plocha

Zdroj: Vlastní zpracování

Při zvolení odkazu „Zákazník“ nás makro přesměruje na formulář „Zákazník“, který vychází z tabulky „Zákazník“. Tento formulář složí k založení nového zákazníka. Zároveň je ve spodní části obrazovky zobrazen výčet veškerých založených zákazníků. Druhé tlačítko pod názvem „Dodavatel“ má stejnou funkci jako tlačítko „Zákazník“. Jediný rozdíl je ve formuláři, který je otevřen. V tomto případě se nám zobrazí formulář „Dodavatel“, který taktéž zobrazuje veškeré informace o dodavatelích, které je možné upravit.

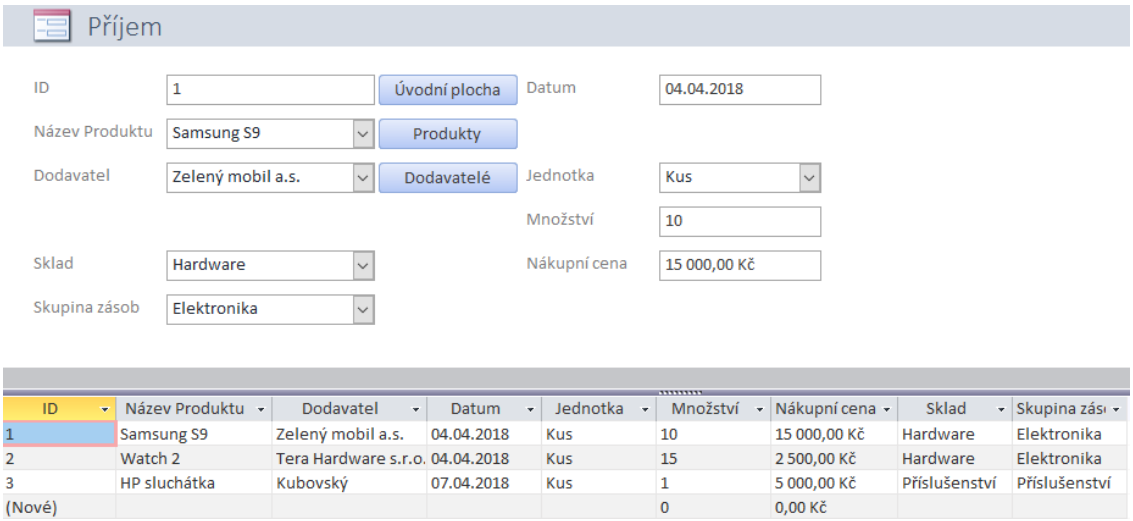


ID	Název	IČO	Adresa	PSČ
1	První sedlecká	72497752	Sedlec	39002
2	Lesy ČR a.s.	12385264	u lesa 5	12345
4	Monitory s.r.o.	45678912	Chelčinda	39001
5	Slepička a.s.	85296374	skalice	15963
6	Slunečnice k.s.	32552562	Slunečná 458	47002
(Nové)				0

Obrázek č. 22: Formulář Zákazník

Zdroj: Vlastní zpracování

Formulář „Příjem“ slouží k příjmu zboží do skladu. Do tohoto formuláře se uživatel dostane z úvodní plochy přes tlačítko „Příjem“. V tomto formuláři je možné zadat informace spojené s příjmem produktů na sklad. Každý příjem má své ID, které jej naprosto přesně definuje. Pokud se jedná o nový produkt, který není v databázi produktů tak se přes tlačítko „Produkty“ přesuneme do formuláře „Produkty“ a zde vytvoříme nový řádek s novým produktem. Stejná operace je potřeba pro případ, když nemáme založeného dodavatele.

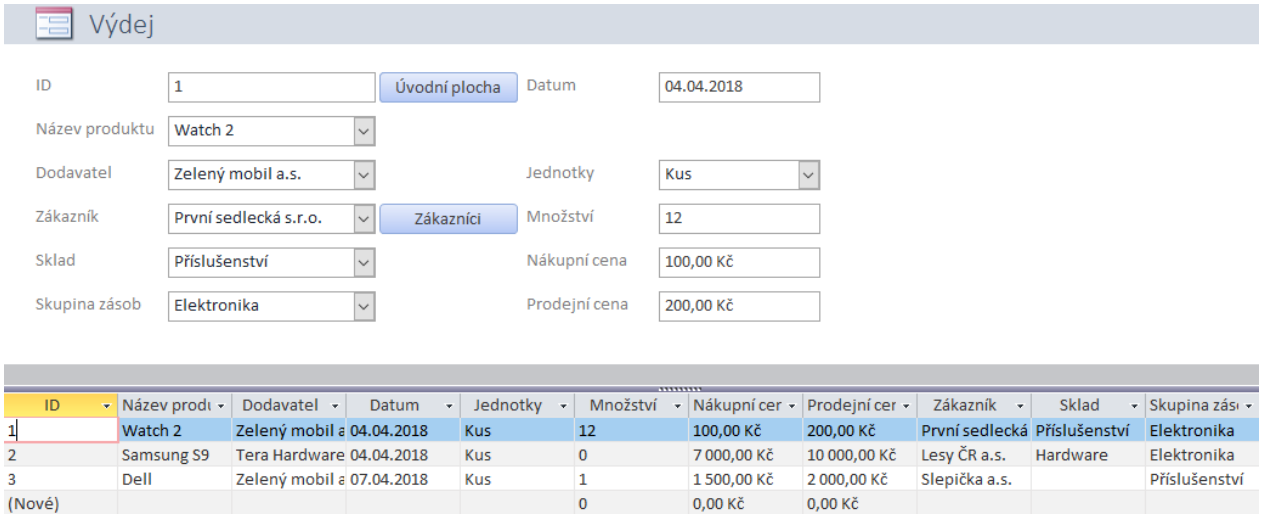


Obrázek č. 23: Formulář Příjem

Zdroj: Vlastní zpracování

Pokud uživatel bude potřebovat vytvořit výdejku zboží ze skladu tak si na úvodní ploše zvolí odkaz „Výdej“ a bude díky makru přesměrován na formulář „Výdej“. Zde bude taktéž automaticky zobrazeno formulářové pole pro zadávání hodnot a ve spodní části je seznam veškerých výdejek ze skladu.

Při vydávání nemá uživatel možnost zakládat nový produkt ani dodavatele. Při výdeji si zvolí ze seznamu produkt, kde uvidí „Název“, aktuální množství a dodavatele produktu.



Obrázek č. 24: Formulář Příjem

Zdroj: Vlastní zpracování

Pomocí dotazu si uživatel může snadno zobrazit aktuální stav zboží na skladě. Toto množství se upravuje při vytváření příjemky nebo výdejky, a to tak že uživatel se odkáže do daného produktu a zde změní množství zboží na skladě.

Nově vytvořená aplikace využívá knihovnu DLL pro bezproblémové fungování. Uživateli stačí když bude mít v počítači nainstalovaný program Access a může snadno evidovat své skladové hospodářství.

3.8 Testování

Pro vývoj aplikace jsem zvolil metodu „Vývoj řízený test“, který jak název napovídá, tak testování probíhalo průběžně při vývoji aplikace. Pokud jsem vyzkoušel spustit nějakou část programu a nastala chyba tak to pro mne znamenalo, že test byl úspěšný a našel chybu.

Zhodnocení

Cíl práce, který stanovil vytvoření funkční aplikace pro skladovou evidenci jsem splnil. Avšak možné rozšíření programu je v budoucí době pravděpodobné. Kdyby se program měl využívat ve skladu potravin tak je potřeba doprogramovat trvanlivost každého produktu. Při využití v technickém průmyslu se nabízí možnost zavedení parametru hmotnost, sériové číslo nebo šarže. Zároveň je vidinou větší automatizace celého programu.

Závěr

V teoretické části bylo hlavním záměrem čtenáře seznámit s hlavním rozdělením informačních systémů. Postupně jsem popisoval rozdělení na účetní, ekonomický a ERP systém. Účetní a ekonomické systémy jsou velice jednoduché a hodně omezené svým určením. Na druhou stranu ERP systémy jsou již mnohem rozsáhlejší a dokáží pojmout potřeby celého podniku. Druhým rozdělením je podle názvu málo známá Holisticko-procesní klasifikace která rozděluje systém do ERP jádra, které řídí vnitřní procesy podniku, CRM zpracovává veškeré informace o klientech, SCM zajišťuje dodavatelský řetězec, který může obsahovat pokročilé plánování a Řídící informační systém. Na závěr jsem sepsal členění ERP systému dle funkčnosti a určení.

V následné části jsem se snažil vystihnout veškeré kroky které je potřeba splnit pro vytvoření nové aplikace založené na databázovém systému. V prvních krocích je zapotřebí sestavit konceptuální model, který obsáhne základní funkcionality budoucího programu. Z konceptuálního modelu následuje tvorba logického modelu pomocí aplikace Toad Data Modeler. V tomto kroku již proběhla normalizace databáze do třetí normální formy. Následujícím krokem bylo vygenerování SQL kódu celé databáze a jeho nahrání do programu Access. Toto nahrání dat je díky importu více formátů velice příjemné.

Posledním krokem bylo zprovoznění aplikace jako celku. Přes makra a VBA jsem vytvořil odkazy mezi formuláři a tabulkami, které zjednoduší budoucímu uživateli ovládání aplikace. Pomocí cizích klíčů jsem zpřístupnil data mezi jednotlivými tabulkami.

Literatura

Amikon.cz. Ekonomický systém na míru Vaší firmě [online]. nedatováno [cit. 2018-01-20]. Dostupné z: http://www.amikon.cz/ekonomicky-system-ucetni-software/

BASL, Josef. Podnikové informační systémy: podnik v informační společnosti. 1. vyd. Praha: Grada, 2002, 142 s. ISBN 80-2470-214-2.

Epadus.cz Ekonomický systém nebo ERP? [online]. 2009, 6. 9. 2009 [cit. 2018-01-20]. Dostupné z: http://epadus.cz/clanky-a-rady/Ucetnictvi/3/.

MOLNÁR, Zdeněk. Efektivnost informačních systémů. 1. vyd. Praha: Grada, 2000, 142 s. ISBN 807169410x.

Orax.cz. Ekonomický a informační systém ORAX [online]. 2018 [cit. 2018-01-20]. Dostupné z: https://www.orax.cz/informacni-system-obecne-informace.

REK, Pavel. Co vás čeká při implementaci IS (seriál 1.díl). Erpforum.cz [online]. 2010, 16 Listopad 2010 [cit. 2018-01-20]. Dostupné z: https://www.erpforum.cz/erp-projekty/co-vas-ceka-pri-implementaci-is-serial-i-1dil.html.

REK, Pavel. Jak na implementaci IS? (seriál 3.díl). Erpforum.cz [online]. 2011, 16 Únor 2011 [cit. 2018-01-20]. Dostupné z: https://www.erpforum.cz/erp-projekty/jak-na-implementaci-is-serial-3dil.html.

ŘEPA, Václav. Analýza a návrh informačních systémů. Vyd. 1. Praha: Ekopress, 1999, 403 s. ISBN 80-86119-13-0.

SODOMKA, Petr a Hana KLČOVÁ. Informační systémy v podnikové praxi: Petr Sodomka, Hana Klčová. 2., aktualiz. a rozš. vyd. Brno: Computer Press, 2010, 501 s. ISBN 978-80-251-2878-7.

SODOMKA, Petr. Aktuální trendy trhu s informačními systémy pro malé a střední podniky. Cvis.cz [online]. 2012, 20.07.2012 [cit. 2018-01-20]. Dostupné z: http://www.cvis.cz/hlavni.php?stranka=novinky/clanek.php&id=1272.

TVRDÍKOVÁ, Milena. Zavádění a inovace informačních systémů ve firmách. 1.vyd. Praha: Grada Publishing, 2000. ISBN 80-716-9703-6.

Co je to DLL?. Podpora Microsoftu [online]. Online, 04.05.2017 [cit. 2018-04-08]. Dostupné z: https://support.microsoft.com/cs-cz/help/815065/what-is-a-dll

ŠIMŮNEK, Milan. SQL kompletní kapesní průvodce [online]. Online: Grada, 2001 [cit. 2018-04-08].

Toad Data Modeler. Quest [online]. Online [cit. 2018-04-08]. Dostupné z: https://www.quest.com/products/toad-data-modeler/

AUER, Liisa. Database Design and System Design. Www2.amk.fi [online]. 2006, 22.8.2006 [cit.2018-01-26]. Dostupné z: http://www2.amk.fi/digma.fi/www.amk.fi/opintojaksot/0303011/1146161367915/1146161680673/1146161836562/1146161929756.html

CONNOLLY, Thomas M. a Carolyn E. BEGG. Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson Education, 2005. ISBN 978-0-321-21025-8.

REK, Pavel. Co vás čeká při implementaci IS (seriál 1.díl). Erpforum.cz [online]. 2010, 16 Listopad 2010 [cit. 2018-03-20]. Dostupné z: https://www.erpforum.cz/erp-projekty/co-vas-ceka-pri-implementaci-is-serial-i-1dil.html.

Summary

The aim of my bachelor thesis is to present the difficulties connected with the draft and the creation of the database application. These complications come from the reality and their detection and understanding are not so easy. For their overview and correct understanding the ER Diagrams are used.

The first part of my thesis presents the company information systems. The basic division of the information systems displays accounting information system, economical information system and ERP one. The next way how it is possible to divide the information systems is Holistic-processing classification. This method precisely defines the fundamental concepts ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), SCM (Supply Chain Management), MIS (Management information system). In the second part of the thesis I work on theoretical parts of the database design. Each database comprises the entities, attributes, relations, domain and keys. Without these essential concepts the function of the database would not be possible at all.

The following chapters are focused on creating of templates of the database which is used for the storage agenda of the company. Each of these chapters is targeted on different parts of the database formation and creation, and describes their essential entities, attributes, relations and normalization of data. Creation of this application is displayed from the conceptual, logical and physical model to the creation of the application.

Souhrn

Cílem bakalářské práce je zobrazit obtíže které souvisí s návrhem a tvorbou databázové aplikace. Tyto obtížné situace vycházejí z reality a jejich zachycení a pochopení není zcela jednoduché. Pro jejich zpřehlednění a správné pochopení se využívají ER diagramy.

Úvodní kapitola seznamuje s firemními informačními systémy. V této kapitole je zobrazeno základní rozdělení informačního systému na účetní informační systém, ekonomický informační systém a ERP systém. Dalším způsobem, kterým lze rozdělovat informační systém je Holisticko-procesní klasifikace. Toto rozdělení přesně definuje pojmy ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), SCM (Supply Chain Management), MIS (Management information system). V druhé části se zabývám teoretickými částmi database designu. Každá databáze se skládá z entit, atributů, relací, domén a klíčů. Bez těchto základních prvků by databáze nikdy nefungovaly.

Zbývající kapitoly jsou zaměřeny tvorbě návrhových vzorů databáze skladové agendy podniku. Každá z kapitol se věnuje některé části tvorby databáze a popisuje její nejzásadnější entity, atributy, relace a normalizace dat. Tvorba aplikace je zobrazena od konceptuálního, logického a fyzického modelu až po samotnou tvorbu aplikace.

Key words

CIS	Customer Information System
CRM	Customer Relationship Management
ERP	Enterprise Resource Planning
MIS	Management information system
SCM	Supply Chain Management
ERD	Entity-relationship diagram, diagram entit a vztahů
SQL	Structured Query Language
VBA	Visual Basic for Applications
DLL	Dynamic-link library

Seznam obrázků

Seznam příloh

Příloha A: Toad Data Modeler přiložen na datovém nosiči

Příloha B: SQL kód pro převod do programu Access

Příloha C: Aplikace v Accessu přiložena na datovém nosiči

Příloha B

```
Created: 28.03.2018
' Modified: 28.03.2018
' Model: BP - convert
' Database: MS Access 2007-2010
'=====================================================
'=== Microsoft Access 2000/2002/2003 database creation method
'===
'=== 1. Create a new database in the Microsoft Access
'=== 2. Create a new module
'=== 3. Copy the TDM3 output SQL script into the new Microsoft Access module
'=== 4. Select from main menu "Tools" item "References..." and check
'===    the "Microsoft ActiveX Data Objects 2.x Library"
'===    and "Microsoft ADO Ext. 2.x for DDL and Security"
'===    and "Microsoft DAO 3.6 Object Library"
'=== 5. Place your mouse cursor somewhere in the main procedure Main()
'=== 6. Run the module code (Click the "Run Sub/UserForm" button or press F5)
'=====================================================
Public con As New ADODB.Connection
Public cat As New ADOX.Catalog
Public tbl As ADOX.Table
Public idx As ADOX.Index
Public dbs As DAO.Database
Sub Main()
Set con = CurrentProject.Connection
cat.ActiveConnection = con
Set dbs = CurrentDb()
On Error GoTo ErrorHandler
  Call CreateTables
  Call CreatePrimaryKeys
  Call CreateIndexes
  Call CreateAlterKeys
  Call CreateRelations
  Call CreateRelationsDAO
  Call CreateQueries
  MsgBox "Script successfully processed.", vbInformation
  Exit Sub
ErrorHandler:
  Select Case Err.Number
    Case -2147217857 'DAO: 3010
      MsgBox "Table " & tbl.Name & " already exist!", vbInformation
      Err.Clear
    Case -2147217868 'DAO: 3284
      MsgBox "Index " & idx.Name & " for table " & tbl.Name & " already exist!", vbInformation
      Err.Clear
```

```
    Case Else
      MsgBox Err.Description, vbCritical
  End Select
End Sub
' Create tables
'===============
Sub CreateTables()
Call CreateTable1 'Zákazník
Call CreateTable2 'Produkty
Call CreateTable3 'Dodavatelé
Call CreateTable4 'Pohyby
Call CreateTable5 'Skladová místa
Call CreateTable6 'Jednotky
Call CreateTable7 'Přepočty jednotek
Call CreateTable8 'Položky tabulky přepočtových jednotek
Call CreateTable9 'Typ skladového pohybu
End Sub
'=== Create table Zákazník ========
Sub CreateTable1()
Set tbl = New ADOX.Table
tbl.Name = "Zákazník"
cat.Tables.Append tbl
Call AddFieldToTable("Zákazník", "Název", adVarWChar, 10, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Zákazník", "IČO", adInteger, 0, 0, "", "FALSE", "8", "", "", "", "")
Call AddFieldToTable("Zákazník", "Adresa", adVarWChar, 20, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Zákazník", "PSČ", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Zákazník", "Kontaktní osoba", adVarWChar, 1, 0, "", "TRUE", "", "", "", "", "")
Call AddFieldToTable("Zákazník", "ID zákazníka", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
dbs.TableDefs.Refresh
Call AddPropertyToFieldDAO( "Zákazník","Název","Caption","Attribute1",dbText)
Call AddPropertyToFieldDAO( "Zákazník","IČO","Caption","Attribute2",dbText)
Call AddPropertyToFieldDAO( "Zákazník","Adresa","Caption","Attribute3",dbText)
Call AddPropertyToFieldDAO( "Zákazník","PSČ","Caption","Attribute4",dbText)
Call AddPropertyToFieldDAO( "Zákazník","Kontaktní osoba","Caption","Attribute5",dbText)
Call AddPropertyToFieldDAO( "Zákazník","ID zákazníka","Caption","Attribute6",dbText)
End Sub
'=== Create table Produkty ========
Sub CreateTable2()
Set tbl = New ADOX.Table
tbl.Name = "Produkty"
cat.Tables.Append tbl
Call AddFieldToTable("Produkty", "ID", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Produkty", "Název", adVarWChar, 1, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Produkty", "Měrná jednotka", adVarWChar, 1, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Produkty", "Aktuální stav", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Produkty", "Nákupní cena", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
```

```
Call AddFieldToTable("Produkty", "Prodejní cena", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Produkty", "Tabulka přepočtu", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
dbs.TableDefs.Refresh
Call AddPropertyToFieldDAO( "Produkty","ID","Caption","Attribute6",dbText)
Call AddPropertyToFieldDAO( "Produkty","Název","Caption","Attribute1",dbText)
Call AddPropertyToFieldDAO( "Produkty","Měrná jednotka","Caption","Attribute2",dbText)
Call AddPropertyToFieldDAO( "Produkty","Aktuální stav","Caption","Attribute5",dbText)
Call AddPropertyToFieldDAO( "Produkty","Nákupní cena","Caption","Attribute3",dbText)
Call AddPropertyToFieldDAO( "Produkty","Prodejní cena","Caption","Attribute4",dbText)
Call AddPropertyToFieldDAO( "Produkty","Tabulka přepočtu","Caption","Attribute7",dbText)
End Sub
'=== Create table Dodavatelé ========
Sub CreateTable3()
Set tbl = New ADOX.Table
tbl.Name = "Dodavatelé"
cat.Tables.Append tbl
Call AddFieldToTable("Dodavatelé", "ID", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Dodavatelé", "Dodavatel", adVarWChar, 20, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Dodavatelé", "IČO", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Dodavatelé", "Adresa", adVarWChar, 20, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Dodavatelé", "PSČ", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Dodavatelé", "Kontaktní osoba", adVarWChar, 15, 0, "", "TRUE", "", "", "", "", "")
dbs.TableDefs.Refresh
Call AddPropertyToFieldDAO( "Dodavatelé","ID","Caption","Attribute1",dbText)
Call AddPropertyToFieldDAO( "Dodavatelé","Dodavatel","Caption","Attribute2",dbText)
Call AddPropertyToFieldDAO( "Dodavatelé","IČO","Caption","Attribute3",dbText)
Call AddPropertyToFieldDAO( "Dodavatelé","Adresa","Caption","Attribute4",dbText)
Call AddPropertyToFieldDAO( "Dodavatelé","PSČ","Caption","Attribute5",dbText)
Call AddPropertyToFieldDAO( "Dodavatelé","Kontaktní osoba","Caption","Attribute6",dbText)
End Sub
'=== Create table Pohyby ========
Sub CreateTable4()
Set tbl = New ADOX.Table
tbl.Name = "Pohyby"
cat.Tables.Append tbl
Call AddFieldToTable("Pohyby", "Typ pohybu", adVarWChar, 1, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "Datum", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "Produkt", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "Množství", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "Nákupní cena", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "Prodejní cena", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "ID skladu", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "Doklad", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "Kontaktní osoba", adVarWChar, 15, 0, "", "TRUE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "ID zákazník", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Pohyby", "ID dodavatel", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
```

```
dbs.TableDefs.Refresh
Call AddPropertyToFieldDAO( "Pohyby","Typ pohybu","Caption","Attribute1",dbText)
Call AddPropertyToFieldDAO( "Pohyby","Datum","Caption","Attribute2",dbText)
Call AddPropertyToFieldDAO( "Pohyby","Produkt","Caption","Attribute3",dbText)
Call AddPropertyToFieldDAO( "Pohyby","Množství","Caption","Attribute5",dbText)
Call AddPropertyToFieldDAO( "Pohyby","Nákupní cena","Caption","Attribute6",dbText)
Call AddPropertyToFieldDAO( "Pohyby","Prodejní cena","Caption","Attribute7",dbText)
Call AddPropertyToFieldDAO( "Pohyby","ID skladu","Caption","Attribute8",dbText)
Call AddPropertyToFieldDAO( "Pohyby","Doklad","Caption","Attribute9",dbText)
Call AddPropertyToFieldDAO( "Pohyby","Kontaktní osoba","Caption","Attribute10",dbText)
Call AddPropertyToFieldDAO( "Pohyby","ID zákazník","Caption","Attribute11",dbText)
Call AddPropertyToFieldDAO( "Pohyby","ID dodavatel","Caption","Attribute11",dbText)
End Sub
'=== Create table Skladová místa ========
Sub CreateTable5()
Set tbl = New ADOX.Table
tbl.Name = "Skladová místa"
cat.Tables.Append tbl
Call AddFieldToTable("Skladová místa", "ID skladu", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Skladová místa", "Název skladu", adVarWChar, 30, 0, "", "FALSE", "", "", "", "", "")
dbs.TableDefs.Refresh
Call AddPropertyToFieldDAO( "Skladová místa","ID skladu","Caption","Attribute1",dbText)
Call AddPropertyToFieldDAO( "Skladová místa","Název skladu","Caption","Attribute2",dbText)
End Sub
'=== Create table Jednotky ========
Sub CreateTable6()
Set tbl = New ADOX.Table
tbl.Name = "Jednotky"
cat.Tables.Append tbl
Call AddFieldToTable("Jednotky", "Název jednotky", adVarWChar, 20, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Jednotky", "Zkratka jednotky", adVarWChar, 5, 0, "", "FALSE", "", "", "", "", "")
dbs.TableDefs.Refresh
Call AddPropertyToFieldDAO( "Jednotky","Název jednotky","Caption","Attribute1",dbText)
Call AddPropertyToFieldDAO( "Jednotky","Zkratka jednotky","Caption","Attribute2",dbText)
End Sub
'=== Create table Přepočty jednotek ========
Sub CreateTable7()
Set tbl = New ADOX.Table
tbl.Name = "Přepočty jednotek"
cat.Tables.Append tbl
Call AddFieldToTable("Přepočty jednotek", "ID hlavičky", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Přepočty jednotek", "Název tabulky přepočtových jednotek", adVarWChar, 20, 0,
"", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Přepočty jednotek", "ID", adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
dbs.TableDefs.Refresh
Call AddPropertyToFieldDAO( "Přepočty jednotek","ID hlavičky","Caption","Attribute1",dbText)
```

```
Call AddPropertyToFieldDAO( "Přepočty jednotek","Název tabulky přepočtových
jednotek","Caption","Attribute2",dbText)
Call AddPropertyToFieldDAO( "Přepočty jednotek","ID","Caption","Attribute6",dbText)
End Sub
'=== Create table Položky tabulky přepočtových jednotek ========
Sub CreateTable8()
Set tbl = New ADOX.Table
tbl.Name = "Položky tabulky přepočtových jednotek"
cat.Tables.Append tbl
Call AddFieldToTable("Položky tabulky přepočtových jednotek", "ID řádku přepočtové tabulky",
adInteger, 0, 0, "", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Položky tabulky přepočtových jednotek", "Základní jednotka", adInteger, 0, 0, "",
"FALSE", "", "", "", "", "")
Call AddFieldToTable("Položky tabulky přepočtových jednotek", "Vyšší jednotka", adInteger, 0, 0, "",
"FALSE", "", "", "", "", "")
Call AddFieldToTable("Položky tabulky přepočtových jednotek", "Faktor", adSingle, 0, 0, "", "FALSE", "",
"", "", "", "")
Call AddFieldToTable("Položky tabulky přepočtových jednotek", "ID hlavičky", adInteger, 0, 0, "",
"FALSE", "", "", "", "", "")
Call AddFieldToTable("Položky tabulky přepočtových jednotek", "Zkratka jednotky", adVarWChar, 5, 0,
"", "FALSE", "", "", "", "", "")
Call AddFieldToTable("Položky tabulky přepočtových jednotek", "ID", adInteger, 0, 0, "", "FALSE", "", "",
"", "", "")
dbs.TableDefs.Refresh
Call AddPropertyToFieldDAO( "Položky tabulky přepočtových jednotek","ID řádku přepočtové
tabulky","Caption","Attribute1",dbText)
Call AddPropertyToFieldDAO( "Položky tabulky přepočtových jednotek","Základní
jednotka","Caption","Attribute2",dbText)
Call AddPropertyToFieldDAO( "Položky tabulky přepočtových jednotek","Vyšší
jednotka","Caption","Attribute3",dbText)
Call AddPropertyToFieldDAO( "Položky tabulky přepočtových
jednotek","Faktor","Caption","Attribute4",dbText)
Call AddPropertyToFieldDAO( "Položky tabulky přepočtových jednotek","ID
hlavičky","Caption","Attribute1",dbText)
Call AddPropertyToFieldDAO( "Položky tabulky přepočtových jednotek","Zkratka
jednotky","Caption","Attribute2",dbText)
Call AddPropertyToFieldDAO( "Položky tabulky přepočtových
jednotek","ID","Caption","Attribute6",dbText)
End Sub
'=== Create table Typ skladového pohybu ========
Sub CreateTable9()
Set tbl = New ADOX.Table
tbl.Name = "Typ skladového pohybu"
cat.Tables.Append tbl
Call AddFieldToTable("Typ skladového pohybu", "ID pohyb", adInteger, 0, 0, "", "FALSE", "", "", "", "",
"")
Call AddFieldToTable("Typ skladového pohybu", "Název pohybu", adVarWChar, 20, 0, "", "FALSE", "",
"", "", "", "")
Call AddFieldToTable("Typ skladového pohybu", "Příjem", adVarWChar, 1, 0, "", "FALSE", "", "", "", "",
"")
```

```
Call AddFieldToTable("Typ skladového pohybu", "Výdej", adVarWChar, 1, 0, "", "FALSE", "", "", "", "", "")
dbs.TableDefs.Refresh
Call AddPropertyToFieldDAO( "Typ skladového pohybu","ID pohyb","Caption","Attribute1",dbText)
Call AddPropertyToFieldDAO( "Typ skladového pohybu","Název
pohybu","Caption","Attribute2",dbText)
Call AddPropertyToFieldDAO( "Typ skladového pohybu","Příjem","Caption","Attribute3",dbText)
Call AddPropertyToFieldDAO( "Typ skladového pohybu","Výdej","Caption","Attribute4",dbText)
End Sub
' Create primary keys
'=====================
Sub CreatePrimaryKeys()
'=== Create primary key for table Zákazník ======
Set idx = New ADOX.Index
Set tbl = cat.Tables("Zákazník")
idx.Name = "Key1"
idx.PrimaryKey = True
idx.Unique  = True
idx.IndexNulls = adIndexNullsDisallow
Call AddFieldToIndex(idx, "ID zákazníka", False )
tbl.Indexes.Append idx
'=== Create primary key for table Produkty ======
Set idx = New ADOX.Index
Set tbl = cat.Tables("Produkty")
idx.Name = "Key3"
idx.PrimaryKey = True
idx.Unique  = True
idx.IndexNulls = adIndexNullsDisallow
Call AddFieldToIndex(idx, "ID", False )
tbl.Indexes.Append idx
'=== Create primary key for table Dodavatelé ======
Set idx = New ADOX.Index
Set tbl = cat.Tables("Dodavatelé")
idx.Name = "Key4"
idx.PrimaryKey = True
idx.Unique  = True
idx.IndexNulls = adIndexNullsDisallow
Call AddFieldToIndex(idx, "ID", False )
tbl.Indexes.Append idx
'=== Create primary key for table Pohyby ======
Set idx = New ADOX.Index
Set tbl = cat.Tables("Pohyby")
idx.Name = "Key5"
idx.PrimaryKey = True
idx.Unique  = True
idx.IndexNulls = adIndexNullsDisallow
Call AddFieldToIndex(idx, "Produkt", False )
```

```
tbl.Indexes.Append idx
'=== Create primary key for table Skladová místa ======
Set idx = New ADOX.Index
Set tbl = cat.Tables("Skladová místa")
idx.Name = "Key6"
idx.PrimaryKey = True
idx.Unique  = True
idx.IndexNulls = adIndexNullsDisallow
Call AddFieldToIndex(idx, "ID skladu", False )
tbl.Indexes.Append idx
'=== Create primary key for table Jednotky ======
Set idx = New ADOX.Index
Set tbl = cat.Tables("Jednotky")
idx.Name = "Key7"
idx.PrimaryKey = True
idx.Unique  = True
idx.IndexNulls = adIndexNullsDisallow
Call AddFieldToIndex(idx, "Zkratka jednotky", False )
tbl.Indexes.Append idx
'=== Create primary key for table Přepočty jednotek ======
Set idx = New ADOX.Index
Set tbl = cat.Tables("Přepočty jednotek")
idx.Name = "Key8"
idx.PrimaryKey = True
idx.Unique  = True
idx.IndexNulls = adIndexNullsDisallow
Call AddFieldToIndex(idx, "ID hlavičky", False )
Call AddFieldToIndex(idx, "ID", False )
tbl.Indexes.Append idx
'=== Create primary key for table Položky tabulky přepočtových jednotek ======
Set idx = New ADOX.Index
Set tbl = cat.Tables("Položky tabulky přepočtových jednotek")
idx.Name = "Key9"
idx.PrimaryKey = True
idx.Unique  = True
idx.IndexNulls = adIndexNullsDisallow
Call AddFieldToIndex(idx, "ID řádku přepočtové tabulky", False )
Call AddFieldToIndex(idx, "ID hlavičky", False )
Call AddFieldToIndex(idx, "Zkratka jednotky", False )
Call AddFieldToIndex(idx, "ID", False )
tbl.Indexes.Append idx
'=== Create primary key for table Typ skladového pohybu ======
Set idx = New ADOX.Index
Set tbl = cat.Tables("Typ skladového pohybu")
idx.Name = "Key10"
idx.PrimaryKey = True
```

```
idx.Unique  = True
idx.IndexNulls = adIndexNullsDisallow
Call AddFieldToIndex(idx, "ID pohyb", False )
tbl.Indexes.Append idx
End Sub
' Create indexes
'================
Sub CreateIndexes()
End Sub
' Create alternate keys (unique indexes in Microsoft Access)
'=====================================================
Sub CreateAlterKeys()
End Sub
' Create relations
'==================
Sub CreateRelations()
Dim keyFk As ADOX.Key
'=== Create relationship between parent table Přepočty jednotek and child table Položky tabulky
přepočtových jednotek ======
Set keyFk = New ADOX.Key
Set tbl = cat.Tables("Položky tabulky přepočtových jednotek")
keyFk.Name = "Relationship5"
keyFk.Type = adKeyForeign
keyFk.RelatedTable = "Přepočty jednotek"
Call AddFieldToRelation(keyFk, "ID hlavičky", "ID hlavičky")
Call AddFieldToRelation(keyFk, "ID", "ID")
tbl.Keys.Append keyFk
'=== Create relationship between parent table Jednotky and child table Položky tabulky přepočtových
jednotek ======
Set keyFk = New ADOX.Key
Set tbl = cat.Tables("Položky tabulky přepočtových jednotek")
keyFk.Name = "Relationship8"
keyFk.Type = adKeyForeign
keyFk.RelatedTable = "Jednotky"
Call AddFieldToRelation(keyFk, "Zkratka jednotky", "Zkratka jednotky")
tbl.Keys.Append keyFk
'=== Create relationship between parent table Jednotky and child table Položky tabulky přepočtových
jednotek ======
Set keyFk = New ADOX.Key
Set tbl = cat.Tables("Položky tabulky přepočtových jednotek")
keyFk.Name = "Relationship9"
keyFk.Type = adKeyForeign
keyFk.RelatedTable = "Jednotky"
Call AddFieldToRelation(keyFk, "Zkratka jednotky", "Zkratka jednotky")
tbl.Keys.Append keyFk
'=== Create relationship between parent table Produkty and child table Přepočty jednotek ======
Set keyFk = New ADOX.Key
```

```
Set tbl = cat.Tables("Přepočty jednotek")
keyFk.Name = "Relationship11"
keyFk.Type = adKeyForeign
keyFk.RelatedTable = "Produkty"
Call AddFieldToRelation(keyFk, "ID", "ID")
tbl.Keys.Append keyFk
End Sub
' Create relations (DAO)
'=======================
Sub CreateRelationsDAO()
Dim rel As DAO.Relation
dbs.Relations.Refresh
End Sub
' Create queries
'================
Sub CreateQueries()
Dim cmd As ADODB.Command
End Sub
' Add fields to table
'====================
Sub AddFieldToTable(TableName As String, FieldName As String, _
        DataType As Integer, SizePrecCol As Integer, ScaleCol As Integer, Attributes As String, _
        Nullable As String, DefaultValue As Variant, _
        AllowZeroLength As String, CompressUnicode As String, _
        ValText As String, ValRule As String)
Dim col As New ADOX.Column
col.ParentCatalog = cat
col.Name = FieldName
col.Type = DataType
If DataType = adVarWChar Then col.DefinedSize = SizePrecCol
If DataType = adNumeric Then
  col.Precision = SizePrecCol
  col.NumericScale = ScaleCol
End If
If Nullable <> "" Then col.Properties("Nullable").Value = CBool(Nullable)
If Attributes <> "" Then col.Properties(Attributes).Value = True
If AllowZeroLength <> "" Then col.Properties("Jet OLEDB:Allow Zero Length").Value =
CBool(AllowZeroLength)
If CompressUnicode <> "" Then col.Properties("Jet OLEDB:Compressed UNICODE Strings").Value =
CBool(CompressUnicode)
If DefaultValue <> "" Then col.Properties("Default").Value = DefaultValue
If ValRule <> "" Then col.Properties("Jet OLEDB:Column Validation Rule").Value = ValRule
If ValText <> "" Then col.Properties("Jet OLEDB:Column Validation Text").Value = ValText
cat.Tables(TableName).Columns.Append col
End Sub
' Add DAO properties to table
'============================
```

```
Sub AddPropertyToTableDAO( TableName As String, PropertyName As String, Value As Variant,
DataType As String)
Dim prp As DAO.Property
Dim tdf As DAO.TableDef
Set tdf = dbs.TableDefs(TableName)
Set prp = tdf.CreateProperty(PropertyName, DataType, Value)
tdf.Properties.Append prp
End Sub
' Add DAO properties to field
'============================
Sub AddPropertyToFieldDAO( TableName As String, FieldName As String, PropertyName As String,
Value As Variant, DataType As String)
Dim prp As DAO.Property
Dim fld As DAO.Field
Dim tdf As DAO.TableDef
Set tdf = dbs.TableDefs(TableName)
Set fld = tdf.Fields( FieldName )
Set prp = fld.CreateProperty(PropertyName, DataType, Value)
fld.Properties.Append prp
End Sub
' Add fields to index
'====================
Sub AddFieldToIndex(idx As ADOX.Index, FieldName As String, Descending As Boolean )
idx.Columns.Append FieldName
If Descending = True Then idx.Columns(FieldName).SortOrder = adSortDescending
End Sub
' Add fields to relation
'========================
Sub AddFieldToRelation(keyFk As ADOX.Key, PKField As String, FKField As String )
keyFk.Columns.Append FKField
keyFk.Columns(FKField).RelatedColumn = PKField
End Sub
' Add fields to relation (DAO)
'============================
Sub AddFieldToRelationDAO( rel As DAO.Relation, PKField As String, FKField As String )
Dim fld As DAO.Field
Set fld = rel.CreateField( PKField )
fld.ForeignName = FKField
rel.Fields.Append fld
End Sub
```