VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ

ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT

INSTITUTE OF INFORMATICS

NÁVRH DATABÁZE PRO NEZISKOVOU ORGANIZACI

PROPOSAL OF DATABASE FOR A NON-PROFIT ORGANIZATION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — RADEK ROZBOŘIL

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

Rozbořil Radek

Manažerská informatika (6209R021)

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

Návrh databáze pro neziskovou organizaci

v anglickém jazyce:

Proposal of Database for a Non-profit Organization

Pokyny pro vypracování:

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

Podle § 60 zákona č. 121/2000 Sb. (autorský zákon) v platném znění, je tato práce "Školním dílem". Využití této práce se řídí právním režimem autorského zákona. Citace povoluje Fakulta podnikatelská Vysokého učení technického v Brně.

Seznam odborné literatury:

BEGG, C. E., R. HOLOWCZAK a T. CONOLLY. 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.
GROFF, J. R. a P. N. WEINBERG. SQL: kompletní průvodce. 1. vyd. Brno: CP Books, 2005. 936 s. ISBN 80-251-0369-2.
MOLINARO, A. SQL: kuchařka programátora. 1. vyd. Brno: Computer Press, 2009. 573 s. ISBN 978-80-251-2617-2.
STANEK, W. R. Microsoft SQL Server 2012: kapesní rádce administrátora. 1. vyd. Brno: Computer Press, 2013. 544 s. ISBN 978-80-251-3797-0.

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 zabývá návrhem databáze pro neziskovou organizaci Diakonie ČCE Betlém. Výsledný návrh by měl zlepšit, a hlavně zjednodušit, pracovní podmínky v organizaci. Návrh vychází z analýzy prostředí a z požadavků organizace.

ABSTRACT

Bachelor thesis deals with the database design for a non-profit organization Diakonie ČCE Betlém. The resulting design should improve and simplify working conditions in the organization. The project is based on an analysis of the organization's environment and the requirements of the organization.

KLÍČOVÁ SLOVA

SQL, databáze, návrh databáze, nezisková organizace

KEYWORDS

SQL, database, database design, non-profit organization

BIBLIOGRAFICKÁ CITACE

ROZBOŘIL, R. *Návrh databáze pro neziskovou organizaci.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2015. 61 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. 121/2000Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 25. května 2015 ……

podpis studenta

PODĚKOVÁNÍ

Rád bych poděkoval mému vedoucímu práce, panu Ing. Janu Luhanovi, Ph.D., za konzultace a za čas, který věnoval mým dotazům. Také děkuji panu Ing. Kačenovi za cenné rady a svým rodičům, kteří mě při studiu podporovali.

OBSAH

ÚVOD

S databázemi se každý z nás setkává denně a staly se tak nedílnou součástí naší společnosti. Jedná se o určitý systém uspořádání důležitých dat např. evidence čtenářů v knihovně, studentů na vysoké škole či zaměstnanců v nějaké firmě.

Organizace, které se pohybují ve stejném sektoru nabízených produktů či služeb, se od sebe můžou lišit v používání softwarových programů, určených pro spolehlivý chod společnosti. I přes velké množství produktů, které se na trhu vyskytují, jsou někdy firmy nuceni vytvořit si vlastní systém „na míru“ organizace. Nabízené programy totiž nemusí obsahovat to, co firma vyžaduje.

Téma databázové systémy jsem si vybral, protože se mi naskytla příležitost navrhnout databázi na reálném objektu. O databázích jsem získával teoretické znalosti během studia a díky této příležitosti si můžu své znalosti vyzkoušet v praxi.

Databáze bude přehledná a jednoduchá, ale i tak by měla splňovat požadavky organizace. U samotného návrhu bude kladen důraz především na její funkčnost. Celková práce by měla vést k návrhu databáze, která má za úkol zlepšit efektivitu práce zaměstnanců v organizaci.

CÍL A METODIKA PRÁCE

Hlavním cílem bakalářské práce je návrh databáze pro neziskovou organizaci Diakonie ČCE středisko Betlém, sídlící na jižní Moravě v Kloboukách u Brna. Aktuální forma evidence organizaci již nevyhovuje, a tak se rozhodla pro vytvoření nového způsobu ukládání dat. Návrh musí odpovídat požadavkům organizace. Konkrétně databáze musí vést evidenci dárců a jejich darů a rozdělení těchto složek do jednotlivých skupin či podskupin.

První část práce je zaměřená na popis teoretických východisek, které jsou potřeba pro tvorbu praktické části. V druhé části se nachází základní informace o společnosti a analýza současného stavu organizace.

Třetí, praktická část obsahuje samotné vytvořené návrhu databáze, která splňuje jak požadavky organizace, tak požadavky současných norem. Hlavní návrh bude doplněn (pro účely zefektivnění databáze) i o část fyzického návrhu, ve které budou procedury, trigger (spoušť) a pohledy. Logický návrh je proveden v programu Case Studio 2 a fyzický návrh databáze v programu Microsoft SQL Server 2012.

1 TEORETICKÁ VÝCHODISKA

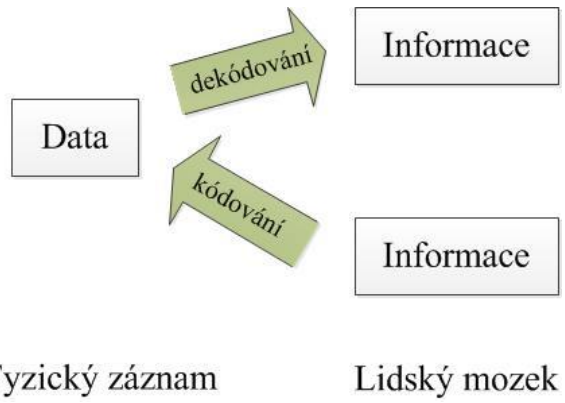
Tato kapitola je zaměřená na základní pojmy a na teoretická východiska užívaná při tvorbě a návrhu databází.

1.1 Základní pojmy

1.1.1 Data

Data můžeme chápat jako nezpracované fakta, která mají nějakou důležitost pro jednotlivce nebo organizaci. (1)

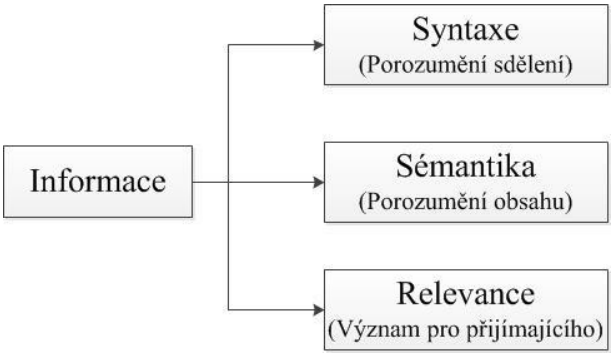
Samotná data může člověk uložit, zpracovat nebo je transformovat do jiné podoby (např. zapsáním na papír, do databáze). (2)



Obr. 1: Vztah dat s informacemi (Zdroj: 2)

1.1.2 Informace

Můžeme říci, že se jedná o data, která prošla zpracováním a dostala nějakou strukturu. Daná informace nám dává nějaký význam. Informace získáváme např. z dat, které byly uloženy do databáze. Informace splňuje dle obrázku č. 2 tři požadavky. (1)



Obr. 2: Informace (Zdroj: 2)

1.1.3 Znalosti

Znalost můžeme chápat jako informaci o využití dat a jiných informací. Jednodušeji si znalost lze vysvětlit na příkladu, kdy jsme svědky dopravní nehody. Tuto znalost máme díky tomu, že jsme již nějaké dopravní nehody dříve viděli. Následně situaci můžeme vyhodnotit okamžitou reakcí, kdy tuto nehodu nahlásíme záchranným složkám (hasiči, zdravotnické služby) nebo vyprostíme zraněné osoby z automobilu a provedeme základní ošetřením. (2)

1.2 Datové modely

1.2.1 Lineární datový model

Mezi objekty modelu není žádná vazba ani žádný vztah. Každý záznam má pouze svého předchůdce a následovníka. (2)

1.2.2 Hierarchický datový model

U tohoto modelu je vztah nadřízenosti a podřízenosti. Můžeme říci, že se jedná o vztah záznamů typu rodič-potomek, tedy každá složka má svoji podsložku. (2)

1.2.3 Síťový datový model

Podobně jako u hierarchického modelu i zde platí vztah nadřízenosti a podřízenosti různých tabulek. Ovšem tady může mít jeden potomek více rodičovských tabulek. Tyto relace (spojené rodičovské tabulky s dceřinou tabulkou) se někdy nazývají jako sady. (2)

1.2.4 Relační datový model

Jeden z nejpoužívanějších modelů současnosti. Jelikož se budu zabývat relačním modelem více, věnuji tomuto modelu samostatnou kapitolu.

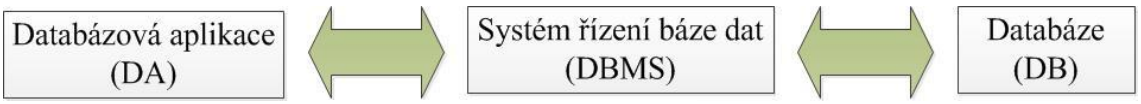
1.2.5 Objektový datový model

Nejnovější model, který je vystavěný na základním prvku, tedy objektu. Objekt v sobě navíc zahrnuje metody, které určují jeho chování. Každý objekt má svůj identifikátor (OID), díky kterému můžeme vést mezi objekty vazby. (2)

1.3 Databázový systém

Databázový systém (zkráceně DS) je kolekce aplikací. Dohromady je tvořen ze tří částí, které spolu navzájem interagují. Konkrétně se jedná o části: (1)

- Databázová aplikace (DA)
- Systém řízení báze dat (DBMS)
- Databáze (DB) (1)



Obr. 3: Databázový systém (Zdroj: vlastní zpracování)

1.3.1 Databázová aplikace

Počítačový program, který uživatelé využívají pro vytvoření či správu databáze. Tato aplikace vyvolá příkaz jazyka SQL pro následné DBMS. (1)

1.3.2 Systém řízení báze dat (DBMS)

Systém, ve kterém může uživatel vytvářet a udržovat databázi. Samotný systém také poskytuje řízený přístup k vytvořené databázi. Dále DBMS umožňuje vkládat, třídit, udržovat, aktualizovat a mazat data v databázi. Podporuje používat dotazovací jazyk SQL. (1)

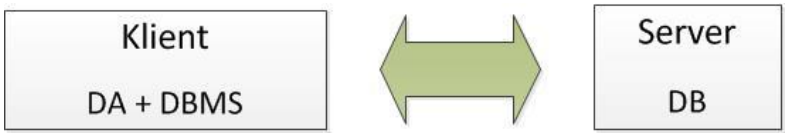
Zajímavou vlastností je také řízení zotavení, tedy navrácení databáze před stav, kdy vznikla nějaká nežádoucí operace (např. selhání transakce, havárie disku). Dále poskytuje mechanizmy pro zabezpečení dat (neautorizovaný přístup). V neposlední řadě zabezpečuje integritu dat v databázi. Rozlišujeme také různé druhy architektury DBMS. (3)

1.3.2.1 Jednovrstvá architektura

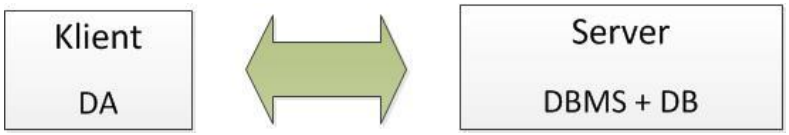
Někdy označovaná jako centrální architektura. Jedná se o starší typ, který byl využíván u sálových počítačů (mainframe). Na centrálním PC se nachází logika provozu, zpracování dat, databáze a datové služby. Přístup byl umožněn přes „neinteligentní“ terminály, které se skládaly pouze ze zobrazovací jednotky (monitor), klávesnice a hardwaru, který zajišťoval komunikaci s centrálním počítačem. (3)

1.3.2.2 Dvouvrstvá architektura

Často označovaná jako architektura **klient/server**, která se využívá u jazyka SQL. Všechny typy úrovně služeb (DA, DBMS, DB) se rozdělují mezi tyto dvě vrstvy. Lze je rozdělit na dva typy, tedy architektura soustředěná u klienta a architektura soustředěná na serveru. (3)



Obr. 4: Architektura soustředěná u klienta (Zdroj: vlastní zpracování)



Obr. 5: Architektura soustředěná na serveru (Zdroj: vlastní zpracování)

1.3.2.3 Třívrstvá architektura

Každá úroveň služeb má svoji vlastní vrstvu. Příkladem může být případ kdy DA „běží“ na klientovi, DBMS na aplikačním serveru a DB je na serveru databázovém. (3)

Obr. 6: Třívrstvá architektura (Zdroj: vlastní zpracování)

1.3.3 Databáze

Současný pojem databáze vzniká v 60. letech 20. století. Databázi můžeme chápat jako velké uložiště uspořádaných dat pro různá oddělení a uživatele. Data by měla být integrována do databáze bez duplikací. Obvykle vlastní databázi nějaká společnosti, nikoliv jedinec či oddělení. (1)

Databáze v sobě mají také tzv. **metadata**, tedy „data o datech“, které se můžou označovat jako datový slovník. (1) Rozlišujeme několik druhů databází:

1.3.3.1 Papírové databáze

S papírovými databázemi se v dnešním světě už moc nesetkáme, nicméně i zde se jedná o určitý způsob uspořádání důležitých dat. Klasickým příkladem papírové databáze je např. vedení kartotéky pacientů u lékaře. (3)

1.3.3.2 Systémy sálových počítačů (mainframe)

Tato databáze má schopnost ukládat velké objemy dat a je pro ně charakteristická vlastnost dobrého připojení. Nevýhodou jsou ovšem rozměry těchto počítačů. (3)

1.3.3.3 Souborově orientované databáze (dBase)

Využívaly se zhruba do 60. let 20. století. Každá tabulka v databázi používá svůj jeden samostatný soubor. Tato skupina předcházela relačním databázovým systémům. Dají se použít pro menší databáze, kde se předpokládá menší počet přístupů uživatelů. Nevýhodou je ovšem značná duplikace dat. (3)

1.3.3.4 Relační databázové systémy

Oproti souborově orientovaným databázím mají relační databáze lepší datovou integritu. V dBase je za integritu odpovědný aplikační program, zatím co v relačních databázových systémech se datová integrita přesouvá do samotné databáze. (3)

1.3.3.5 Objektově orientované databáze

Principem je, že data neukládáme do několika tabulek, ale uložíme je dohromady jako objekt s vlastnostmi, které dále udržujeme. Představuj tak jiný pohled na data. Ukládají se jako XML. (3)

1.4 Relační datový model

Relační model navrhl jako první E. F. Codd. Má za úkol eliminovat z databáze explicitní struktury typu rodič-potomek. Můžeme říci, že relační databáze je taková databáze, kde se veškerá data viditelná pro uživatele uspořádají do tabulky různých datových hodnot. Všechny databázové operace pak pracují s těmito relacemi. (4)

1.4.1 Model dat

Jedná se o integrovanou kolekci konceptů, která nám slouží pro popis jednotlivých dat, relací mezi daty a omezení dat, které jsou používané organizací. Dá se říci, že model je reprezentace „reálného světa“ událostí a objektů. Model nám zachycuje kromě dat i vzájemné vazby. (1)

1.4.2 Terminologie

Základním prvkem modelu je relace. Zjednodušeně se dá říci, že relace je jedna databázová tabulka, která má jedinečný název a je složená z řádků a sloupců. Využívá se k ukládání informací o objektech. V terminologii můžeme definovat pět základních klíčových položek: (1)

Relace – tabulka se sloupci a řádky.

Atribut – pojmenovaný sloupec relace.

Datová n-tice – řádek relace.

Doména – množina přípustných hodnot pro jeden a více atributů

Relační databáze – kolekce normalizovaných tabulek. (1)

1.4.3 Vlastnosti relačních tabulek

Každá tabulka musí mít své jméno, které je jedinečné a v databázi tabulka s takovým jménem ještě nenachází. Jméno tabulky by nemělo obsahovat mezery a interpunkční znaménka. Každý sloupec musí mít také své unikátní, jedinečné jméno. Hodnoty v jednom sloupci pochází ze stejné domény. Pořadí těchto sloupců v relačních tabulkách není důležité. V relačních tabulkách dále neexistují duplicitní záznamy, tzn. každý záznam v tabulce je jedinečný. (3)

1.4.4 Integrita relačního modelu

1.4.4.1 Kandidátní klíč

Za kandidátní klíč (candidate key) můžeme považovat každý sloupec v tabulce, který jednoznačně identifikuje záznamy v tabulce. Z množiny prvků v kandidátním klíči vybereme jeden primární klíč. (1)

1.4.4.2 Primární klíč

Je důležité, aby každá tabulka (relace) měla svůj jedinečný primární klíč (primary key). Tato hodnota jednoznačně identifikuje záznamy v tabulce, a hodnota primárního klíče tudíž nesmí být NULL. Nemůže tedy nastat situace, že pro dva řádky v tabulce je hodnota primárního klíče totožná. Primární klíč by neměl být příliš velký (počet bajtů). Jedná se zde o entitní integritu. V praxi se jako primární klíč často používá identifikační číslo (ID). (4)

1.4.4.3 Složený primární klíč

Složený klíč (composite key) vznikne složením dvou nebo více sloupců tabulky. Plní úlohu jednoduchého primární klíč. Kombinace těchto dvou (nebo více) hodnot je tedy také jedinečná. (4)

1.4.4.4 Cizí klíč

Cizí klíč v jedné tabulce je shodný s primárním klíčem v tabulce druhé. Díky cizím klíčům jsou tabulky propojené a ukazují na sebe. Kombinací primárního klíč a cizího

klíč nám vznikne relace mezi tabulkami. Jedná se zde o referenční integritu. Primární a cizí klíč tedy tvoří mezi tabulkami relaci rodič-potomek. (4)

1.4.5 Coddova pravidla

Edgar Frank „Ted“ Codd definoval v roce 1985 celkem 12 pravidel, kterými se databáze řídí, pokud se považuje za relační. Tyto pravidla se stala poloficiální definicí relační databáze. (4)

1.4.5.1 Pravidlo informace

Všechny informace v relační databázi se reprezentují explicitně na logické úrovni a jedním způsobem – hodnotami v tabulkách. (4, str. 82)

1.4.5.2 Pravidlo zaručeného postupu

Musí být zajištěno, aby úplně každý údaj v relační databázi byl logicky přístupný použitím názvu tabulky, hodnoty primárního klíče a názvu sloupce. (4, str. 82)

1.4.5.3 Systematické ošetření prázdných hodnot

Prázdné hodnoty jsou systematicky plně podporovány relačním databázovým řídícím systémem pro reprezentaci chybějících informací a neplatných informací nezávisle na datovém typu. (4, str. 82)

1.4.5.4 Dynamický online katalog založený na relačním modelu

Popis databáze se reprezentuje na logické úrovni stejným způsobem jako běžná data tak, aby se oprávnění uživatelé mohli dotazovat na tato data pomocí stejného relačního jazyku, s jehož pomocí se dotazují na normální pravidla. (4, str. 82)

1.4.5.5 Pravidlo komplexního datového podjazyku

Relační systémy mohou podporovat několik jazyků a různé režimy použití terminálu. Nicméně musí existovat přinejmenším jeden jazyk, jehož příkazy jsou vyjádřitelné, na nějakou dobře definovanou syntaxi, jehož řetězce znaků a tento jazyk komplexní podporuje všechny následující položky: definici dat, definici pohledu, manipulaci s daty, omezení integrity, autorizaci, vymezení transakce. (4, str. 82)

1.4.5.6 Pravidlo aktualizace pohledu

Všechny pohledy, které je teoreticky možné aktualizovat, je rovněž možné aktualizovat systémově. (4, str. 82)

1.4.5.7 Vysokoúrovňové vkládání, aktualizace, odstranění

Schopnost zpracovávat základní relace nebo odvozené relace jako jediný operand se aplikuje nejenom na vyhledávání dat, ale rovněž na vložení, aktualizaci a odstranění dat. (4, str. 82)

1.4.5.8 Fyzická datová nezávislost

Aplikační programy a terminálové aktivity zůstávají logicky nedotčené, kdykoliv jsou provedeny nějaké změny buďto v reprezentacích úložiště nebo přístupových metodách. (4, str. 82)

1.4.5.9 Logická datová nezávislost

Aplikační programy a terminálové aktivity zůstávají logicky neporušené, pokud jsou v základních tabulkách provedeny změny v uchovávání informací jakéhokoliv druhu. (4, str. 83)

1.4.5.10 Nezávislost integrity

Omezení integrity specifické pro jednotlivé relační databáze musí být možné definovat v relačním datovém podjazyku a uložit v katalogu nikoliv v aplikačních programech. (4, str. 83)

1.4.5.11 Distribuční nezávislost

Relační databázový řídící systém má distribuční nezávislost. (4, str. 83)

1.4.5.12 Pravidlo nenarušení

Má-li relační systém nízkoúrovňový jazyk, nelze pomocí tohoto jazyka rušit nebo obcházet pravidla integrity a omezení vyjádřená ve vysokoúrovňovém relačním jazyku. (4, str. 83)

1.5 E-R diagram

E-R diagram (Entity-Relationship Diagram) se obvykle využívá pro návrh struktury relačního datového modelu. Využívá se ovšem také pro znázornění dalších dat, které nemají s databází nic společného (např. algoritmů).

1.5.1 Definice

Entita – Říká nám, která data jsou shromažďována a uchovávána. Můžeme říci, že se jedná o jedinečný název tabulku. (5)

Atribut – Charakteristika entity (sloupec entity). (5)

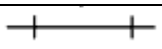
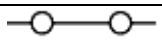
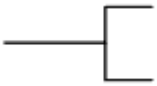
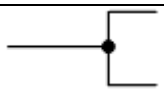
Vazba – Popisuje vazbu (vztah) mezi dvěma entitami. Rozlišujeme čtyři základní vztahy na poměry 1:1, 1:N, N:1 a N:M. Tento poměr nám udává, kolik n-tic relací sobě navzájem odpovídá. (2)

Záznam – Uspořádaný soubor hodnot atributů, které popisují danou entitu. (5)

1.5.2 Symboly

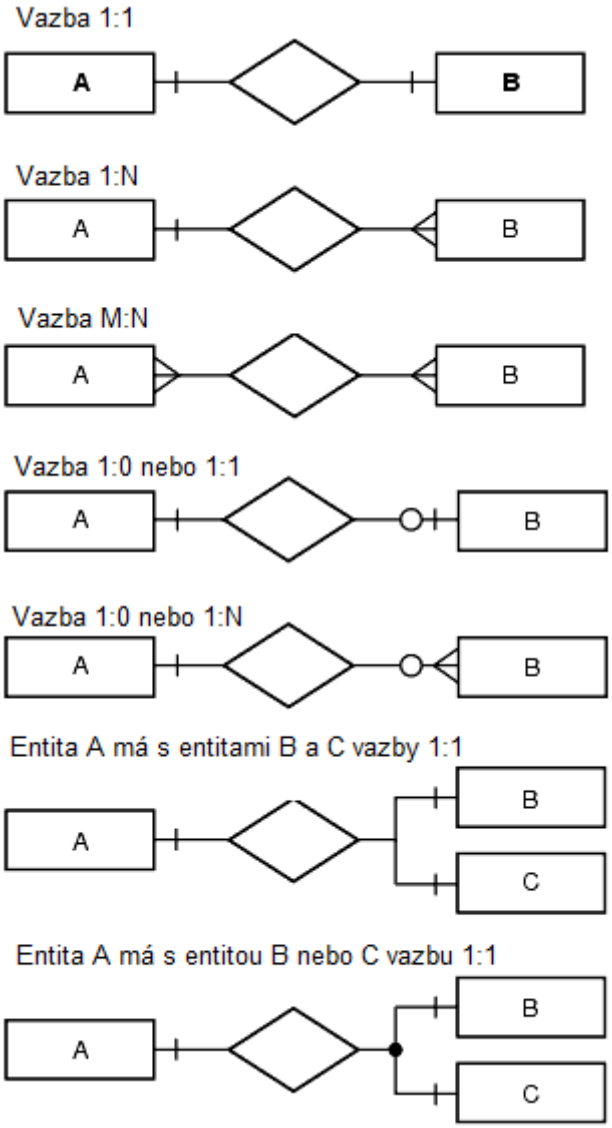
Pro znázornění E-R diagramu se využívají symboly, které jsou uvedeny v následující tabulce.

Tab. 1: Symboly E-R diagramu (Zdroj: upraveno dle 5)

Symbol	Popis
	Typ (název) entity
	Atribut
	Vztah mezi entitami
	Vazba jedna-k-jedné (1:1)
	Vazba jedna-k-více (1:N)
	Vazba více-k-více (M:N)
	Částečně volitelná vazba
	Plně volitelná vazba
	Vzájemně inkluzivní vazba
	Vzájemně se vylučující vazba

1.5.3 Vazby

Následující obrázek znázorňuje příklady základních vazeb mezi entitami.



Obr. 7: Vazby mezi entitami (Zdroj: upraveno dle 5)

1.6 Normalizace

Jedná se o techniku, kterou v roce 1972 vyvinul Edgar F. Codd. Tato technika se provádí jako řada testů na tabulkách v relaci, aby nebyla porušena pravidla pro určitou normální formu. Dohromady existuje několik normálních forem (1NF, 2NF, 3NF, Boyce-Coddova NF, 4NF, 5NF), ovšem nejdůležitější, a hlavně nejvíce využívané, jsou první tři formy. Zjednodušeně můžeme říct, že se jedná o metodu, která má zabránit nadbytečnosti dat v relacích. (1)

Jelikož ve své práci nebudu využívat všechny normální formy, zmíním jen první tři, které jsou pro můj návrh důležité.

1.6.1 První normální forma (1NF)

Tato forma řeší multizávislost. Jedná se o formu, která je důležitá pro vytvoření vhodných tabulek relační databáze. V tabulce se v každém průsečíku sloupce a záznamu vyskytuje jen jediná hodnota, tzn. hodnota nesmí být složená (vícehodnotová). Typickým příkladem složené hodnoty je například adresa. Tuto hodnotu lze do tabulky rozepsat podle jednotlivých položek (ulice, město, PSČ). (1)

1.6.2 Druhá normální forma (2NF)

Tato forma řeší funkční závislost. Tabulka se nachází v druhé normální formě, pokud je v první normální formě a navíc jsou všechny atributy tabulky závislé na primárním klíči. (1)

1.6.2.1 Funkční závislost

Funkční závislost si můžeme představit jako nějaké tvrzení o reálném světě. Například plat zaměstnance závisí na pozici, kterou v podniku vykonává, tj. velikost platu závisí na dané pozici. Celý tento příklad se dá zapsat jako POZICE → PLAT. (2)

1.6.3 Třetí normální forma (3NF)

3NF řeší tranzitivní závislost. Tabulka je ve třetí normální formě, pokud splňuje druhou normální formu a navíc jsou ostatní neklíčové atributy na sebe vzájemně nezávislé. (1)

1.7 Metodologie návrhu databáze

Metodologie návrhu se skládá ze tří částí, které mají za úkol usnadnit proces návrhu a vytvoření databáze. Jednotlivé části jsou dále složeny z několika kroků.

1.7.1 Konceptuální návrh

Proces vytvoření modelu dat používaných v organizaci bez jakýchkoli úvah o fyzické implementaci. (1, str. 206)

V tomto modelu je hlavní úkol vytvoření E-R modelu, který má za úkol představovat přesnou reprezentaci požadavků organizace, kterou má databáze podporovat. Jednoduše můžeme říci, že je potřeba nejprve provést identifikaci entit, relací a spojení atributů s entitami nebo relacemi. Následně se určí doména atributů a určení, které atributy budou kandidátními a následně primárními klíči. Poté se provede kontrola redundance modelu a kontrola, zda výsledný model podporuje uživatelské transakce. (1)

1.7.2 Logický návrh

Proces vytvoření modelu dat používaných organizací, který je založen na specifickém modelu dat, ale nezávislý na konkrétním DBMS a jiných úvahách o fyzické implementaci. (1, str. 206)

V logickém návrhu se „převede“ E-R model z konceptuálního návrhu do množiny relačních tabulek. Struktura je následně zkontrolována pomocí normalizace, aby se minimalizovala redundance. Kontrola probíhá i na jednotlivých tabulkách. Následně jsou definována požadovaná integritní omezení databáze. (1)

1.7.3 Fyzický návrh

Proces vytvoření popisu implementace databáze ve vnější paměti; popisuje podkladové tabulky, organizaci souborů, indexy používané pro dosažení efektivního přístupu k datům, všechna související integritní omezení a bezpečnost omezení. (1, str. 207)

Fyzický návrh obsahuje také několik základních kroků. Nejprve je potřeba převést logický návrh databáze do cílového DBMS. To zahrnuje návrh podkladových tabulek, reprezentace odvozených dat a návrh zbývajících integritních omezení. Následně

organizace zvolí soubory a indexy (analýza transakcí, volba organizace souborů a volba indexů). Poté se provede návrh uživatelských pohledů a bezpečnostních mechanismů. K závěru se provede zvážení zavedení kontrolované redundance a poslední část obsahuje monitorování a doladění systému v samotném provozu. (1)

1.8 Jazyk SQL

První zmínku o jazyku SQL můžeme zaregistrovat v 70. letech 20. století, kdy firma IBM vyvinula standardní jazyk pod názvem SEQUEL. Samotná zkratka SQL značí **Structured Query Language** (strukturovaný dotazovací jazyk). Využívá se jako nástroj pro organizování, správu a získání uložených dat v databázi a je založený na relačním datovém modelu. Zahrnuje v sobě nástroje pro tvorbu tabulek databáze a nástroje na manipulaci s nimi (vložení dat, mazání, vyhledávání, aktualizování). (6)

Jazyk SQL není strukturovaný jazyk v pravém slova smyslu, zejména když jej srovnáme s jazyky, jako jsou Pascal nebo Java. Příkazy v SQL se spíše podobají anglickým větám. Navíc v něm existují některá speciální pravidla, která zabraňují tvorby příkazů, i když vypadají správně a legálně. (4)

Jazyk SQL má v sobě několik různých funkcí:

- Interaktivní dotazovací jazyk
- Databázový programovací jazyk
- Administrační databázový jazyk
- Jazyk aplikací typu klient/server
- Jazyk pro přístup k datům na Internetu
- Distribuovaný databázový jazyk
- Jazyk pro databázové brány (4)

1.8.1 Části jazyka SQL

SQL se skládá z několika částí. Některé jsou určené pro administrátory a návrháře databázových systémů. Další jsou určené pro programátory a koncové uživatele. (7)

1.8.1.1 Data Definition Language

Jazyk pro definici dat (DDL), který slouží převážně pro vytvoření schémat a katalogů v databázi. (7)

1.8.1.2 Storage Definition Language

Zkráceně (SDL). Je používán pro způsob ukládání tabulek. (7)

1.8.1.3 View Definition Language

Třetí část View Definition Languag**e** (VDL) slouží převážně pro návrháře a správce. Zde si můžeme vytvořit například různé pohledy tabulek. (7)

1.8.1.4 Data Manipulation Language

Poslední část jazyka SQL je jazyk Data Manipulation Language (DML), s kterým pracují až koncoví uživatelé databází. Obsahuje čtyři základní příkazy pro manipulaci s daty – SELECT, INSERT, UPDATE, DELETE. (7)

1.8.2 Nativní datové typy SQL

Nativní datové typy jsou veškeré vestavěné typy, které jsou přímo podporované SQL Serverem. Tyto typy mají buď určitou (pevně danou) délku nebo proměnlivou. Délka pro ukládání čísel, tedy u číselných a binárních typů, je představována počtem bajtů. Délka řetězců představuje počet znaků a jsou také omezeny svoji maximální velikostí. Každý svůj typ má také svůj rozsah a také přesnost. Dále rozlišujeme přesnost, která nám značí celkový počet číslic v čísle (počet číslic vpravo od desetinné čárky). (8)

Uvedu jen datové typy, které považuji při mém návrhu za důležité.

1.8.2.1 Čísla a mněna

Celočíselné typy

Bit – hodnoty 0,1 nebo null, velikost 1 bajt.

Integer (int) – -2^{31} (-2 147 483 648) až $2^{31} - 1$ (2 147 483 647), velikost 4 bajty.

Small integer (smallint) – -2^{15} (-32 768) až $2^{15} - 1$ (32 767), velikost 2 bajty.

Tiny integer (tinyint) – 0 až 255, velikost 1 bajt. (8)

Měna

Money - -922 337 203 685 477,5808 až +922 337 203 685 477,5807, velikost 8 bajtů. (8)

Desetinná čísla

Decimal – $-10^{38} + 1$ až $10^{38} - 1$, velikost 5 až 17 bajtů. (8)

1.8.2.2 Časové údaje a znaky

Date – 0001-01-01 až 9999-12-31, velikost 3 bajty.

Time – 00:00:00.0000000 až 23:59:59.9999999, velikost 3 až 5 bajtů.

Datetime – 1. ledna 1753 až 31. prosince 9999, přesnost na tři setiny sekundy, velikost 2 čtyřbajtová celá čísla.

Charakter (char) – znaková data mimo sadu Unicode s pevnou délkou (max 8 000 znaků), velikost 1 bajt na znak.

Character varying (varchar) – znaková data mimo sadu Unicode s proměnnou délkou (do 8 000 znaků), velikost 1 bajt na znak.

Text – znaková data mimo sadu Unicode s proměnnou délkou (do 2 147 483 647 znaků), velikost 1 bajt na znak. (8)

1.8.3 Index

Indexy poskytují rychlejší přístup k datům bez nutnosti procházet celou databázi. Zvyšují výkonost vyhledávání jednotlivých údajů v databázi. (9)

Typickou ukázkou, jak vysvětlit indexy v běžném životě, může být příklad z knihovny. Pokud chceme najít nějakou knihu, stačí prohledat malé kartičky v katalogu, které jsou seřazeny dle názvu. Danou knihu pak díky těmto kartičkám vyhledáme rychleji, než kdybychom ji hledali v několika policích. (9)

Můžeme vytvořit dva druhy indexů – primární a sekundární. U primárního indexu je záruka, že hodnoty záznamu budou jedinečné. Pro vytvoření se využívá SQL příkaz *CREATE UNIQUE INDEX*. V některých programech se primární index vytváří automaticky nad primárním klíčem tabulky. U sekundárního indexu může každé pole odpovídat více záznamů, tzn. hodnoty nemusí být jedinečné Poskytuje mechanismy pro zadání dodatečného klíče pro podkladovou tabulku a tak dochází k efektivnějšímu vyhledávání dat. Vytvoří se příkazem *CREATE INDEX*. (1)

SQL server déle podporuje dva typy indexů – clusterované a neclusterované. Clusterovaný index se může v tabulce vyskytovat jen jednou. Vytváří se za normálních okolností nad primárním klíčem. Hodnoty by zde měli být jedinečné. Neclusterovaných indexů může být v tabulce naopak více (konkrétně až 999). Hodnoty naopak nemusí být jedinečné. (8)

1.8.4 Příkazy SQL

Jazyk SQL dohromady tvoří zhruba 40 příkazů. Daný příkaz vyžaduje, aby databázový řídící systém provedl určitou akci, kterou uživatel požaduje (např. vytvoření nové tabulky). Každý příkaz začíná slovesem a dále pokračují klauzulemi, které začínají klíčovými slovy. Nejzákladnější příkazy jsou uvedeny v následující tabulce. (4)

Tab. 2: Příkazy jazyka SQL (Zdroj: 4)

Příkaz	Popis
Manipulace s daty	
SELECT	Vyhledá data z databáze
INSERT	Vloží do databíze nové řádky dat
DELETE	Odstraní z databáze řádky dat
UPDATE	Modifikuje stávající data databáze
Definice dat	
CREATE TABLE	Vloží novou tabulku do databáze
DROP TABLE	Odstraní tabulku z databáze
ALTER TABLE	Změní strukutru stávající tabluky
CREATE VIEW	Vloží nový pohled do databáze
DROP VIEW	Odstraní pohled z databáze
Řízení transakcí	
COMMIT	Ukončí aktuální transakci
ROLLBACK	Zruší aktuální transakci
Programové SQL	
DECLARE	Definuje kurzorovou tabulku pro dotaz
OPEN	Otevře kurzorovou tabulku
FETCH	Načte řádek výsledků dotazu
CLOSE	Zavře kurzorovou tabulku
EXECUTE	Dynamicky provede příkaz v jazyce SQL

Obr. 8: Příkaz jazyka SQL (Zdroj: 4)

2 ANALÝZA SOUČASNÉHO STAVU

Cílem této kapitoly je seznámit se s neziskovou organizací Betlém, která patří pod organizaci Diakonie ČCE. Dále popíši aktuální systém, který organizace využívá pro evidenci dárců, a požadavky investora na vytvoření návrhu databáze. V poslední řadě jsou zde zmíněny formy evidencí dárců u dalších neziskových organizací.

2.1 Základní informace

Diakonie ČCE – středisko Betlém bylo založeno 7. 3. 1990 v Kloboukách u Brna, a je jedním z prvních středisek Diakonie Českobratrské církve evangelické. Středisko je evidováno jako právnická osoba a je zapsáno v registru poskytovatelů sociálních služeb dle zákona č. 108/2006 Sb., o sociálních službách. (10)

Hlavní náplní této organizace je tedy poskytování sociální služeb. Konkrétněji se dá říci, že poskytuje domov a péči dospělým lidem a dětem s vážným zdravotním postižením. (10)

Středisko Betlém tvoří celkem čtyři pracoviště:

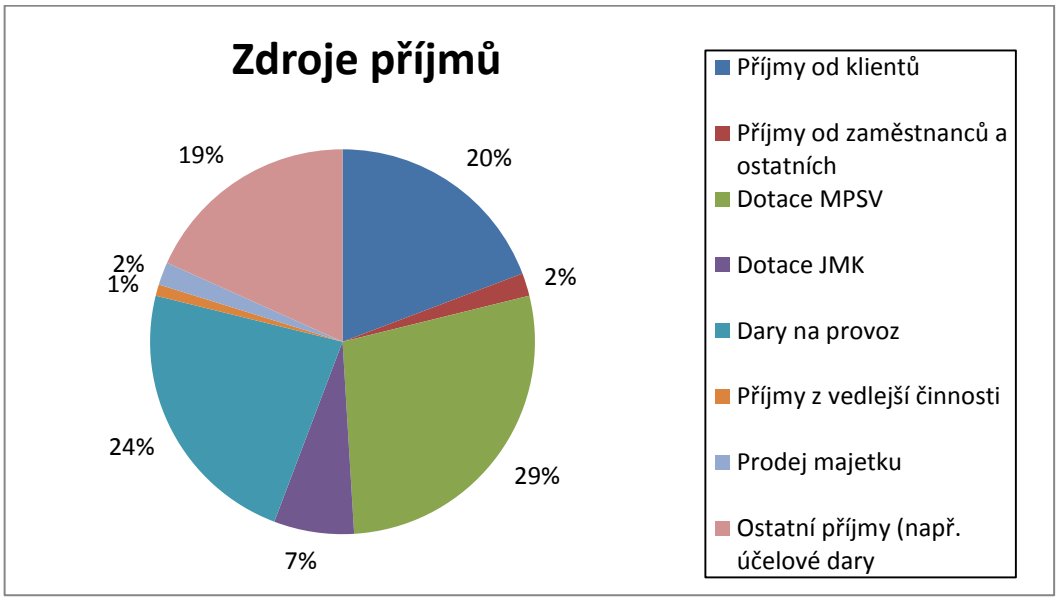
- Domov Betlém v Kloboukách u Brna
- Domov Narnie v Morkůvkách
- Domov Arkénie v Brumovicích
- Domov Mirandie v Brumovicích (10)

Činnost těchto čtyř pracovišť zajišťuje útvar právě v Kloboukách u Brna. Všichni zaměstnanci, klienti a dárci jsou evidováni do svých skupin na jednom centrálním místě.

2.1.1 Financování organizace

Důležitým zdrojem financování organizace jsou příspěvky od dárců, které jsou následně využívány pro zlepšení kvality nabízených služeb a zajištění lepší péče klientům. Finance jsou také využívány na opravy či stavby budov nebo ke koupi vybavení, na které stát organizaci nepřispívá. (10)

Největším zdrojem příjmů jsou dotace, a následně druhým největším zdrojem financování jsou právě dárci. Dárci, kteří organizaci poskytují finanční nebo hmotný majetek, tedy hrají ve financování důležitou roli. Organizace by bez nich prakticky nemohla fungovat. Věnované dary si následně může dárce odečíst ze základu daně. (10)



Graf 1: Zdroje příjmů (Zdroj: 10)

2.2 Uživatelé databáze pro evidenci dárců

Jelikož se jedná o databázi, která je určena pro evidenci dárců a jejich darů, bude ji využívat převážně jedna hlavní složka v organizaci. Jedná se o oddělení fundraisingu.

Fundraising můžeme doslovně přeložit jako „navyšování fondů či zdrojů“. Hlavní úlohou fundraisingu je tedy získání nových dobrovolníků (firem), kteří můžou poskytnout organizaci dary. Má za úkol také zajistit dostatečné zdroje (finanční zdroje, zázemí, lidi) pro fungování organizace. Důležité je též najít co nejvíce zdrojů financovaní, aby nebyla organizace závislá jen na jednom zdroji. Oddělení fundraisingu může také naplánovat různé akce (např. den otevřených dveří), kam může pozvat své stálé, nové, či potencionální dárce. (11)

2.3 Aktuální evidence dárců

Nezisková organizace Betlém sice funguje již mnoho let, ale i přesto nevlastní speciální program, který je určený pro evidenci dárců. Jednotliví dárci jsou složitě ukládáni do papírové kartotéky a do tabulek vytvořených v MS Excel.

2.3.1 Kartotéka

Jedná se o papírovou formu databáze, kdy je každému dárci založena právě jedna složka. Ta obsahuje všechny důležité informace (např. jméno, IČO, kontaktní údaje,…). Pokud existuje, je ve složce také uložena oboustranně podepsaná darovací smlouva. Do složky jsou též vkládány údaje o poskytnutých darech (např. částka, věcný dar a jeho vyjádření v peněžních jednotách, datum obdržení,…).

Tento způsob ukládání dat je v dnešní době již zastaralý. Hlavní nevýhodou této evidence je dlouhé vyhledávání různých údajů. Dále může pracovník udělat při založení nové karty chybu a následné přepisování (škrtání) těchto údajů je nevzhledné. Papír, na kterém je dárce evidován, je samozřejmě omezen svojí velikostí a tak časté přepisování údajů může vést k potřebě přepsání všech údajů do nové karty. Dá se ovšem říci, že kartotéka je evidována organizací spíše pro jakousi archivaci a zálohu dat.

2.3.2 MS Excel

Evidence je také prováděná v MS Excel. Vše je uložené v tabulce, která je rozdělená na název firmy (popř. jméno osoby), kontaktní údaje, datum obdržení daru, částku (případně věcný dar a jeho cenu). V poznámce je dále uvedeno, zda je mezi organizací a dárcem podepsaná darovací smlouva a zda bylo odeslané potvrzení a poděkování o přijatém daru.

Jak je na první pohled zřejmé, v prostředí MS Excel jsou uloženy stejné údaje jako v papírové kartotéce. Jedna věc je tedy prováděna dvakrát a práce tak není efektivní. Vyhledávání potřebných údajů není teda prováděno v papírové kartotéce, ale právě v prostředí MS Excel. Zapisování do tabulek je složité a zabere tak pracovníkům fundraisingu více času, než by potřebovali k evidenci v nové databázi navrhnuté „na míru“.

2.4 Vztah organizace – dárce

Existují dva způsoby, jak se může fyzická či právnická osoba stát dárcem. První možností je, že někdo sám nabídne jako dar nějakou věc nebo peněžní částku z vlastní iniciativy. Druhý způsob je takový, že oddělení fundraisingu kontaktuje různé firmy a podnikatele s žádostí o nějakou pomoc. Samozřejmě je jisté, že některé kontaktované osoby či firmy výzvu na dárcovství odmítnou. Proto je nutné uchovávat i informace o tom, kteří jedinci již byli o dárcovství požádáni a odmítli ho.

Novému dárci je následně vystavena karta, do které se zapíší následující důležité informace v závislosti na tom, jestli se jedná o fyzickou osobu nebo o organizaci: *jméno, příjmení, název firmy* (popř. i *středisko*)*, IČO, DIČO, kontaktní adresa, telefonní číslo* a *e-mail.* Dárci se také přiřadí *variabilní symbol*, pod kterým bude posílat veškeré peněžní dary posílané přes bankovní účet. Následně jsou v *poznámce* vypsány informace, jestli je uzavřena mezi dárcem a organizací oboustranná darovací smlouva a zda-li dárce požaduje po organizaci roční potvrzení. V poslední řadě jsou u dárce evidované jeho poskytnuté *dary.*

U jednotlivých darů jsou ukládány informace: kdo poskytl dar, tedy *dárce*, *datum*, kdy byl dar poskytnut, *typ daru* (hmotný, nehmotný) a jestli byl dar věnován za určitým *účelem.* Pokud je dar hmotný, musí se následně zaznamenat i *vyjádření v peněžních jednotkách*. Dále je zaznamenáno *číslo dokladu*, které potvrzuje přijmutí daru. V *poznámkách* je pak vypsán jeho popis či bližší specifikace. V poslední řadě je evidováno i *variabilní číslo*, jelikož dar může být věnován dárcem, který si nepřeje být zaregistrovaný, a tudíž mu nebude přiděleno unikátní variabilní číslo.

2.5 Formy evidencí u různých neziskových organizací

Kontaktoval jsem několik středisek, které patří pod organizaci Diakonie ČCE. Dle průzkumu jsem zjistil, že každé středisko má svůj vlastní systém evidence. Většina kontaktovaných organizací využívá tabulky v prostředí MS Excel. Některé z nich používají také papírovou formu kartotéky (např. středisko Světlo ve Vrchlabí). Pro evidenci je také možno využít MS Access, který používá např. středisko ve Valašském Meziříčí.

Dále jsem nalezl dva programy, které jsou pro tyto účely evidence vhodné.

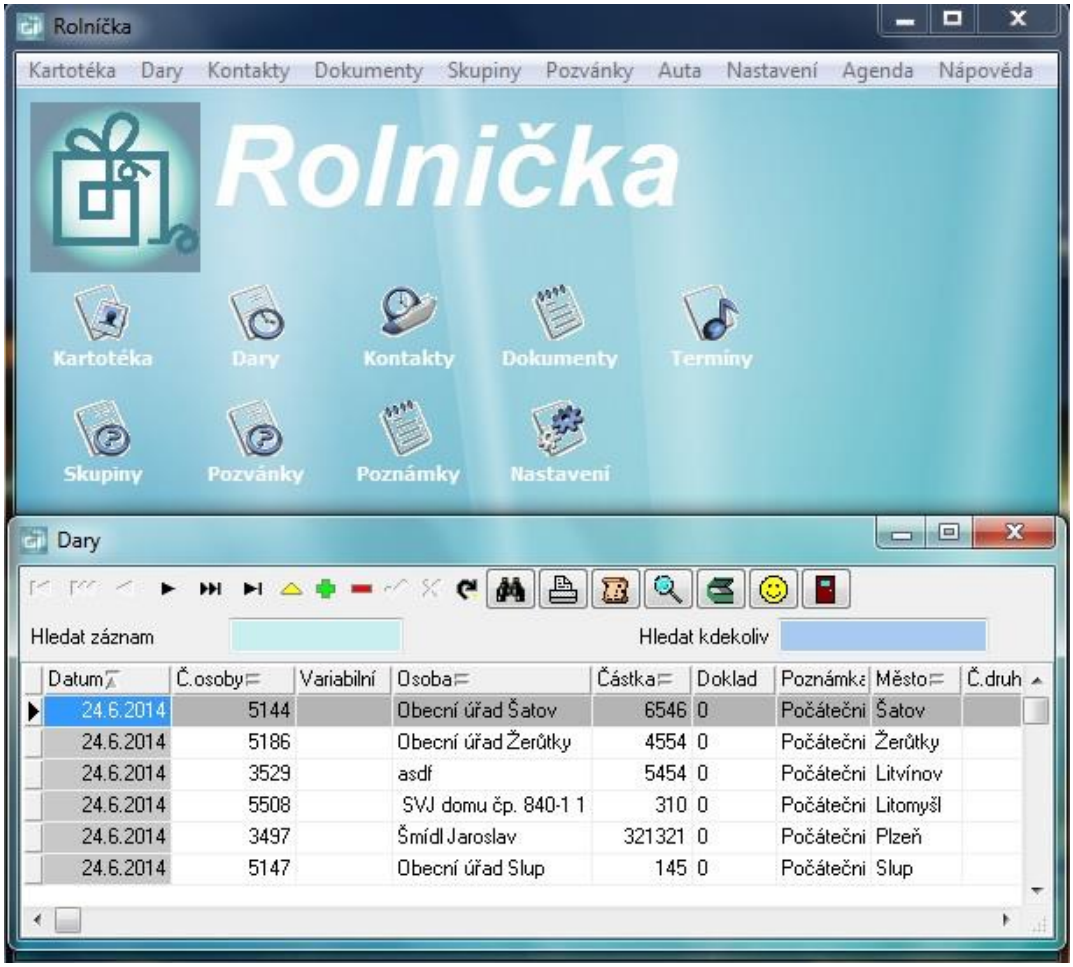
2.5.1 Rolnička

Program vytvořený firmou JIRRA software, s.r.o. Byl vytvořen „na míru“ pro potřeby neziskové organizace Diakonie ČCE - středisko Rolnička. Je určen pro správu dárců, organizací, nadací a různých fyzických a právnických osob, s kterými přichází klient do styku. Umožňuje udržovat jednotlivé dary, kontakty a evidenci různých schůzek a akcí. Taktéž podporuje rozesílání hromadných e-mailů nebo pozvánek. Jednotliví dárci mohou býti rozčleněni dle jednotlivých skupin. Navíc obsahuje evidenci cestovného a knihy jízd. Pořizovací cena (bez omezení počtu záznamu) je 3 900 Kč bez DPH. (12)

Tento program využívají i jiné střediska Diakonie ČCE (např. středisko v Myslibořicích a středisko v Brně).

Firma JIRRA software nabízí na svých internetových stránkách nainstalování „start verze“, díky které si potenciální zákazník může program bezplatně vyzkoušet. Demonstrační program obsahuje vzorová data a není nijak omezen funkčně ani časově. Počet osob v kartotéce je ovšem omezen na 20 záznamů. (12)

Organizace Betlém se již tento program pro evidenci snažila v minulosti zavést. Pro zkoušku zde vytvořili 10 záznamů, ale s postupem času s tímto systémem nakonec nebyli spokojeni a tak od něj upustili. Navíc řeší podstatně více, než organizace požaduje (např. evidenci cestovného).



Obr. 9: Program Rolnička (Zdroj: vlastní zpracování)

2.5.2 Salesforce CRM

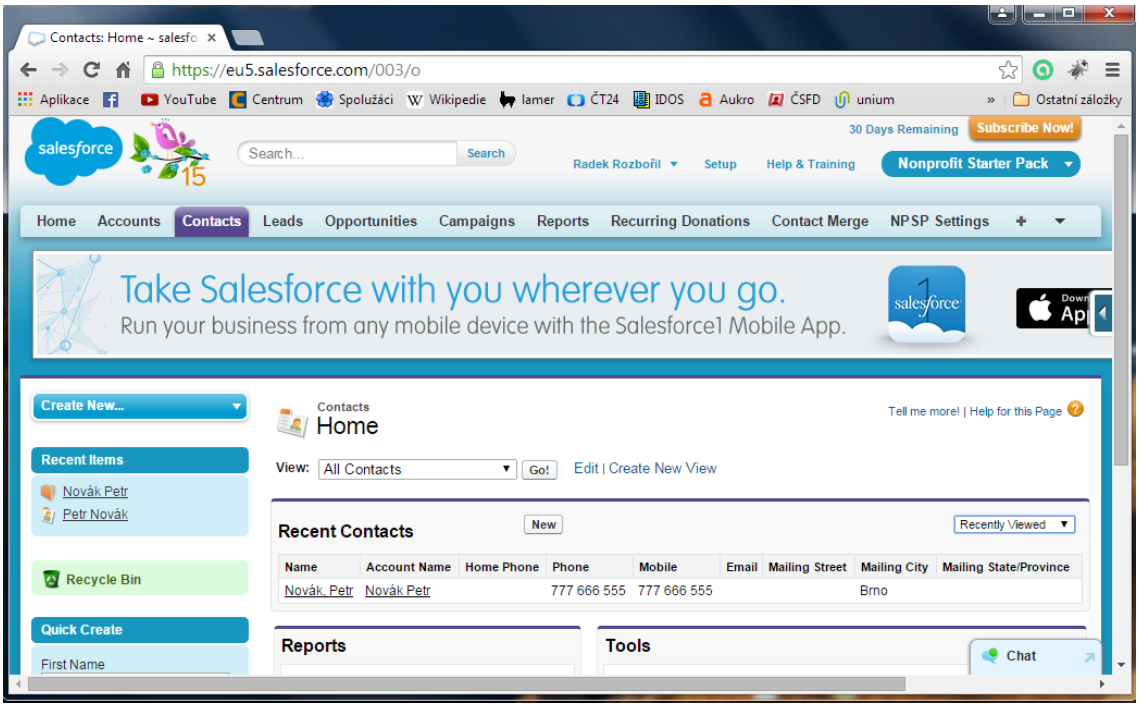
Systém poskytovaný nadací Salesforce. Upravuje se na míru organizacím v neziskovém sektoru pro podporu oddělení fundraisingu, správu projektů a řízení organizace. Takto upravený produkt je někdy nazýván jako „CRM pro nezisovky“. Jeho primární funkcí je především správa vztahů s dárci, partnery a klienty. Dále např. sleduje komunikaci mezi organizací a dárcem, eviduje dary, platby a podporuje rozesílání hromadných emailů apod. (13)

Tento program využívají např. organizace Světlo pro svět – Light for the World, o. s. nebo organizace Čmelák – Společnost přátel přírody. (13)

Program je možný vyzkoušet zdarma bez jakéhokoli omezení po dobu 30 dnů od registrace. Výhodou tohoto programu je, že uživatel nemusí nic instalovat na svůj PC. Aplikace je dostupná prostřednictvím cloud computingu, tzn., data nejsou uloženy

na lokálním PC a vše tedy „běží“ online přes internetový prohlížeč. Nevýhodou je ovšem chybějící česká lokalizace (vše je dostupné pouze v anglickém jazyce). Neziskovým a vzdělávacím organizacím je tento program poskytován zdarma. (13)

Tento systém pro evidenci dárců organizaci nevyhovuje. Někteří uživatelé, kteří potřebují s touto evidencí pracovat, totiž neumí anglický jazyk.



Obr. 10: Systém Salesforce CRM (Zdroj: vlastní zpracování)

2.6 Shrnutí analýzy a požadavků investora

Z analýzy současného stavu organizace a analyzovaných systémů pro evidenci dárců je zřejmé, že tyto systémy organizaci nevyhovují a tudíž je potřeba vytvořit zcela nový systém pro evidenci.

Primárním účelem databáze, kterou investor požaduje, je tedy evidence dárců a jejich darů. Veškerá tato evidence se bude provádět na jednom pracovišti v Klobukách u Brna. U dárců je nutno ukládat všechny důležité informace (jméno, název organizace, IČO,

kontaktní adresa, telefonní číslo,…). Dále chce investor zařadit jednotlivé dárce do různých skupin.

Dary je nutno také rozdělit podle jednotlivých údajů. Je potřeba rozlišovat, zda je dar hmotný nebo nehmotný. Pokud je dar finanční, je důležité evidovat, jestli byly peníze poskytnuty hotově nebo byly odeslány na bankovní účet. Pokud je dar hmotný, je ho nutno vyjádřit i v peněžních jednotkách a dále je požadováno tuto informaci uchovávat. Dar je také možné organizaci věnovat při různých akcích a za různými účely.

Dále chce investor zaznamenat, kam byl dar zařazen, popřípadě jak je s ním dále nakládáno. V poslední řadě je potřeba navrhnout, jací uživatelé budou mít do databáze přístup a jaká budou mít v této databázi práva.

3 NÁVRH ŘEŠENÍ

Ve třetí části se budu zabývat samostatným návrhem databáze. Zmíním zde také jednotlivé tabulky a popíši jejich účel. Vycházet budu především z teoretických poznatků a z předchozí analýzy současného stavu. V poslední řadě se zde nachází ukázky pohledů, trigger a procedury.

3.1 Uživatelé databáze

Dle požadavků organizace definuji skupiny uživatelů, které budou mít přístup do databáze. Každá skupina bude mít odlišná práva. Konkrétně se jedná o skupiny: administrátor, oddělení fundraisingu a PR oddělení.

3.1.1 Administrátor

V organizaci bude tuto úlohu ve skupině zastupovat právě jedna osoba s IT zaměřením. Jedná se o skupinu s nejvyššími přístupovými právy. Závisí na ní veškerý provoz a údržba databáze. Bude mít za úkol také zálohovat data.

3.1.2 Fundraising

V současné době má oddělení fundraisingu 2 zaměstnance. Skupina bude mít právo data zobrazit, vložit nová data (popřípadě upravovat) a mazat je.

3.1.3 PR oddělení

PR (public relations) oddělení je skupina, která nebude mít práva zasahovat do databáze. Uživatelé budou mít možnost data pouze zobrazit.

3.2 Konceptuální návrh

Úkolem bylo navrhnout E-R diagram tak, aby splňoval všechny požadavky organizace. Nejprve jsem určil jednotlivé entity (tabulky) a následně do nich navrhl atributy. Poté bylo nutno vybrat z množiny kandidátních klíčů u každé tabulky jeden primární klíč. V posledním kroku byla provedena kontrola redundance a celého výsledného návrhu.

Díky konceptuálnímu návrhu není pak problém vytvořit návrh logický. Veškeré tabulky jsou popsány v následujícím logickém návrhu.

3.3 Logický návrh

Jak již bylo zmíněno, logický návrh vychází z předešlého konceptuálního návrhu. Celý logický návrh splňuje veškeré požadavky investora. Výsledný E-R diagram je přiložený v příloze č. 1. V příloze č. 2 se pak nachází datový slovník. Návrh byl proveden v programu Case Studio 2.

Celý návrh je možno rozdělit do čtyř částí. První část v sobě uchovává informace o dárci, druhá část pak nese informace o samotném daru. Třetí část databáze je zaměřena na situaci, kdy je dar převeden na majetek organizace. Poslední čtvrtá část zase ukládá, kam byl převeden dar finanční.

3.3.1 Dárce

První hlavní část celé databáze, která se týká uložení všech důležitých dat o dárci. Další tabulky jsou zde kvůli potřebné normalizaci návrhu.

3.3.1.1 Entity

Dárce – hlavní tabulka, v které budou evidování jednotliví dárci. Jako primární klíč zde bude sloužit *ID dárce*. Ukládat se zde bude *jméno* dárce (popř. *název organizace*) a další důležité informace o něm. Bude zde ukládána také *kontaktní adresa* či *telefonní číslo*. Jelikož nechceme mít o dárci uloženy všechny jeho kontakty (ale stačí nám jen jeho hlavní) není potřeba vytvářet novou tabulku určenou jen pro evidenci jednotlivých kontaktů. Dále zde bude uloženo, například jestli je *aktivní*, zda patří do skupiny *VIP*, jestli je mezi organizací a dárcem uzavřená *darovací smlouva*, či zdali je dárce

zaměstnanec nebo klient v organizaci. Každému dárci bude také přidělen *variabilní symbol*.

Ukládat se zde bude i *titul* fyzické osoby a *oslovení*, které bude využíváno při různém kontaktování. U titulů a oslovení by dle normalizace měla být vytvořena tabulka s číselníkem různých možností. Po konzultaci s organizací ale investor nechce využívat u těchto dvou případů číselník a tak zde bude u těchto dvou atributů porušena normalizace.

Dárce středisko – tabulka spojená s hlavní tabulkou Dárce. Řeší možnost, že jeden dárce je velká organizace (firma), rozdělená na několik středisek. I jedno středisko v organizaci tedy může poskytnout dar. Atributy jsou zde pouze *název* střediska, ke kterému se přiřadí *ID*.

Typ dárce – tabulka spojená s hlavní tabulkou Dárce. Rozděluje tři základní typy dárců, kteří jsou: právnická osoba, fyzická osoba a osoba s živnostenským listem. Zde bude tedy ukládán *název* jedné z možností typu dárce.

Rodina – tabulka spojená s hlavní tabulkou Dárce. Řeší možnost, že dar není poskytnut za jednu osoba, ale přímo za celou rodinu (např. rodina Novákova). V tabulce se nachází atributy týkající se *adresy*, jelikož celá rodina může uvést jinou kontaktní adresu, než její zástupce uvedený v tabulce Dárce.

PSČ – tabulka spojená s hlavní tabulkou Dárce a s tabulkou Rodina. Jako primární klíč je zde zvoleno *PSČ*. Dále je možno doplnit *město*, kterému dané PSČ odpovídá.

Skupiny dárců – v této entitě budou uloženy jednotlivé *názvy* skupin, do kterých můžeme dárce rozdělit. Klasickým příkladem je skupina dárců, která požaduje po organizaci zaslat každý rok výroční zprávu.

Skupiny dárců X dárce – entita, spojující tabulku Dárce s tabulkou Skupiny dárců. Díky této tabulce můžeme mít uloženého jednoho dárce v několika různých skupinách. Např. dárce č. 1 je uložen ve skupině, která chce každý rok zaslat výroční zprávu a zároveň i ve skupině, které chce organizace posílat pozvánky na různé akce. Nachází se zde složený primární klíč z atributů *ID dárce* a *ID skupina dárců*.

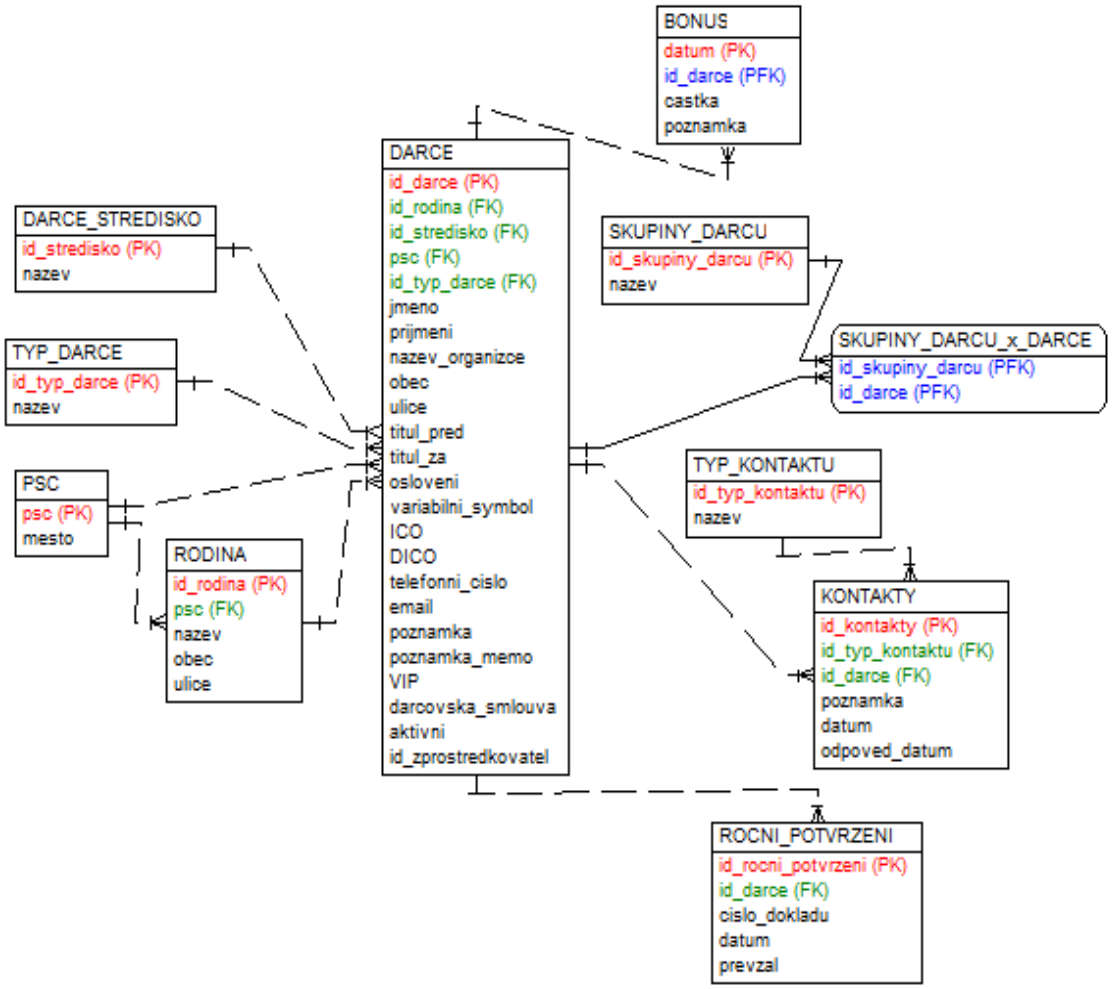
Kontakty – tabulka spojená s tabulkou Dárce. Zde slouží pro evidenci, která zaznamenává, zda organizace svého dárce nějak kontaktovala. Příkladem kontaktu může být zaslaná pozvánka na den otevřených dveří v organizaci. Můžou zde být uloženy i informace o tom, že bylo provedeno kontaktování nového, potencionálního, dárce, který žádost o poskytnutí daru odmítl.

Typ kontaktu – entita spojená s tabulkou Kontakty. Ukládá v sobě informace o tom, jakým způsobem můžeme dárce kontaktovat (např. dopis, e-mail, telefon).

Roční potvrzení – tabulka spojená s tabulkou Dárce. Ukládá v sobě, zda bylo dárci vystaveno roční potvrzení. Dále je uloženo *číslo dokladu*, *datum*, a zda si dárce své potvrzení *převzal*.

Bonus – tabulka spojená s tabulkou Dárce. Řeší možnost, že dárce je klient nebo zaměstnanec v organizaci a následně je této osobě přidělen nějaký bonus v peněžním vyjádření. Klient může dostat např. jednorázovou slevu na pobyt a zaměstnanec naopak bonus k pravidelné mzdě. V *poznámce* bude určen uložen přesný význam bonusu.

3.3.1.2 E-R diagram



Obr. 11: E-R diagram - Dárce (Zdroj: vlastní zpracování)

3.3.2 Dar

Druhá hlavní část celé databáze, která se týká uložení všech důležitých dat o poskytnutém daru. Další tabulky jsou zde také kvůli potřebné normalizaci návrhu.

3.3.2.1 Entity

Dar – druhá hlavní tabulka spojená s tabulkou Dárce. Zde budou ukládány všechny důležité informace o přijatých darech (např. *částka* daru, *číslo dokladu*, *datum*, kdy byl dar poskytnut,…).

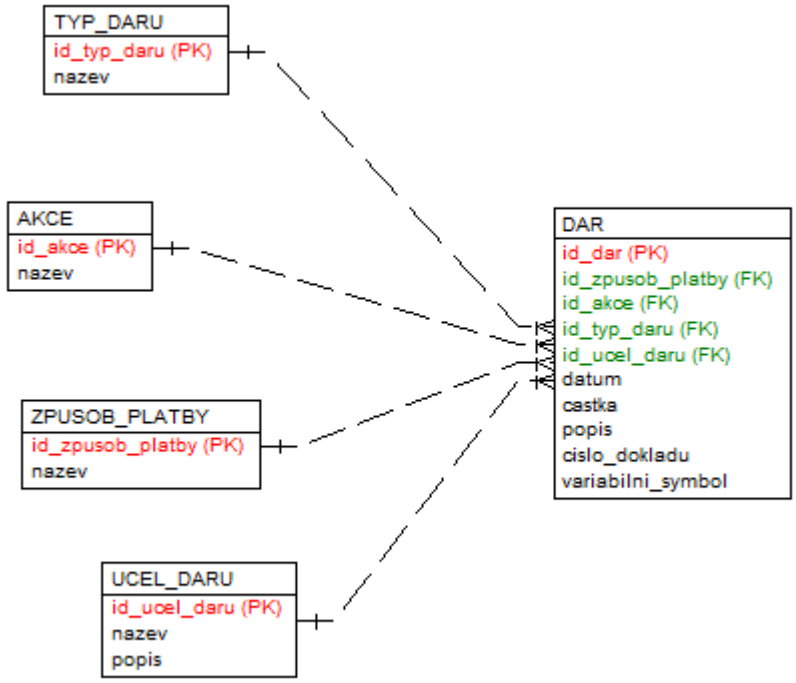
Typ daru – tabulka spojená s tabulkou Dar. Jedná se o číselník, který v sobě bude mít uloženy tři základní *názvy* typu daru, které chce organizace rozlišovat (dar hmotný, nehmotný a finanční).

Akce – tabulka spojená s tabulkou Dar. Entita, v které budou uloženy jednotlivé akce, na kterých byl dar poskytnut. Např. dar mohl být věnován organizaci při vyhlášené veřejné sbírce, při dobročinné aukci apod.

Způsob platby – tabulka spojená s tabulkou Dar. Opět se jedná o číselník, který rozlišuje dva základní způsoby platby (převodem na účet a hotově).

Účel daru – tabulka spojená s tabulkou Dar. Pokud dárce poskytnul svůj dar na předem stanovený účel, budou tyto informace uloženy právě tady. Např. finanční dar je poskytnut na účely koupě nového auta.

3.3.2.2 E-R diagram



Obr. 12: E-R diagram - Dar (Zdroj: vlastní zpracování)

Další entity již neslouží pro evidenci dárců a jejich darů, ale vyskytují se zde pro další potřeby organizace, které plynou z požadavků investora.

3.3.3 Majetek

Poskytnutý dar se stává majetkem organizace. V této části bude ukládáno, kam byl hmotný nebo nehmotný dar přidělen a jak je dál využíván.

3.3.3.1 Entity

Majetek –. Pokud si bude dárce přát v budoucnu vědět, jak se s jeho darem nakládá, budou informace uloženy právě v této tabulce. Nachází se v ní např. *ID daru*, *datum pořízení*, a jeho *cena*.

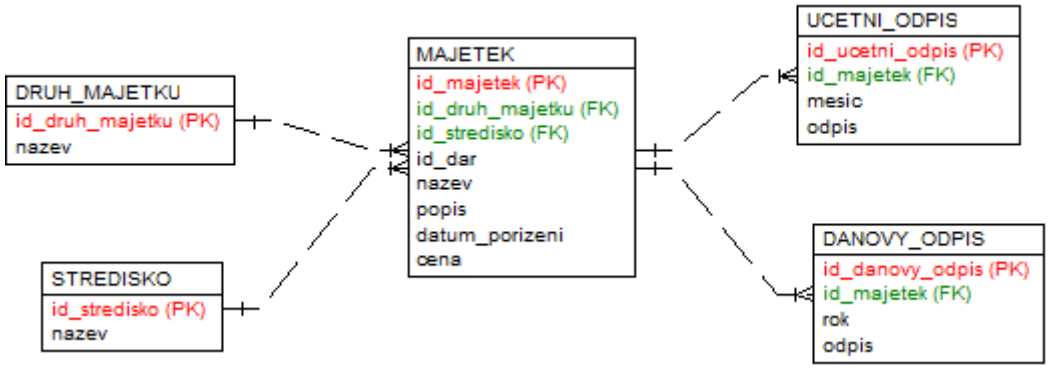
Druh majetku – tabulka spojená s tabulkou Majetek. Rozlišuje dva základní druhy a to hmotný a nehmotný.

Středisko – tabulka spojená s tabulkou Majetek. Zde bude uloženo, jakému středisku byl majetek poskytnut, a kde je také využíván. Organizace Betlém může majetek přidělit dohromady čtyřem střediskům.

Účetní odpis – tabulka spojená s tabulkou Majetek. Ukládá informaci o tom, jaká částka bude odepisována v účetním odpisu majetku.

Daňový odpis – tabulka spojená s tabulkou Majetek. Jedná se o podobný případ jak u předchozí tabulky s tím rozdílem, že se zde ukládá informace o daňovém odpisu majetku.

3.3.3.2 E-R diagram



Obr. 13: E-R diagram - Majetek (Zdroj: vlastní zpracování)

3.3.4 Finanční dar

Pokud jsou organizaci věnované nějaké peníze, bude ukládáno i to, na jaký účet byli peníz převedeny, popřípadě v které pokladně jsou uloženy.

3.3.4.1 Entity

Banka – tabulka, ve které jsou uloženy důležité informace o bance, tedy *název* a *kód banky*. Jako primární klíč zde bude sloužit *ID*. Je spojená s tabulkou Bankovní účet.

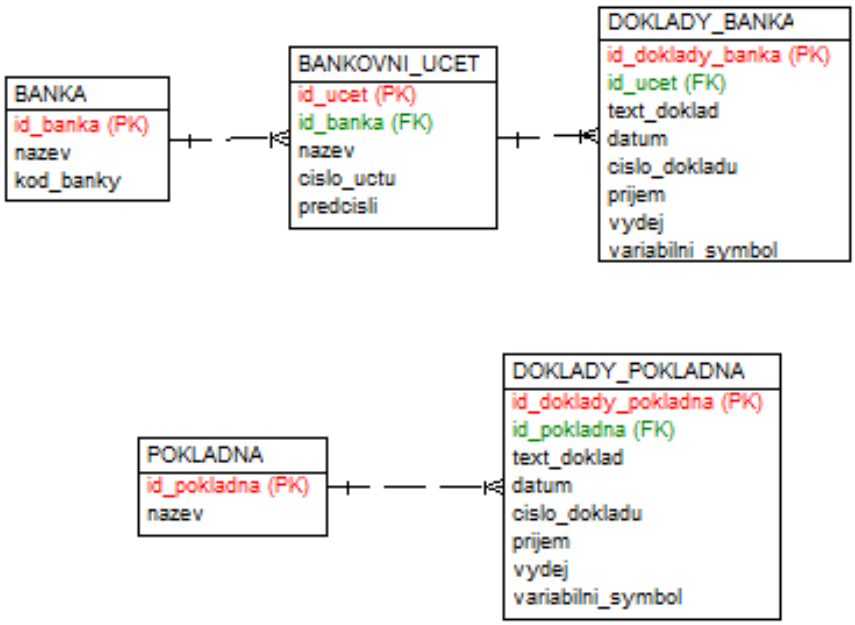
Bankovní účet – ukládá v sobě *název* účtu, jeho *číslo* a *předčíslí*. Jako primární klíč zde slouží *ID*.

Doklady banka – finanční dar, který byl poslán dárcem přes bankovní účet, můžeme nechat zaslat na námi zvolený účet organizace.

Pokladna – obsahuje pouze *název* pokladny a její *ID*.

Doklady pokladna – plní stejnou funkci jako tabulka Doklady banka, s tím rozdílem, že se zde jedná o finanční dary, které byly organizaci předány hotově.

3.3.4.2 E-R diagram



Obr. 14: E-R diagram - Finanční dar (Zdroj: vlastní zpracování)

3.4 Fyzický návrh

Fyzický návrh si bude organizace zajišťovat sama. V této kapitole jsou ovšem popsány procedury, trigger, a pohledy, které je možno využít pro větší efektivitu databáze. Veškerý zdrojový kód je zobrazen v příloze č. 3. Návrh je proveden v prostředí Microsoft SQL Server 2012.

3.4.1 Procedury

O procedurách můžeme říci, že se jedná o nějaký objekt, který je uložen v databázi a vykonává určitou práci s daty.

3.4.1.1 Záloha daru

Tato procedura má jeden vstupní parametr (ID dar). Po zadání tohoto ID se zkopírují z tabulky Dar všechny důležité informace do předem vytvořené tabulky Dary archiv. Pokud se v databázi nenachází dar se zadaným ID, vytiskne procedura hlášku o špatném zadání vstupního parametru. Zkouška dotazu na spuštění procedury je spuštěná přes transakci, která je následně odvolaná.

```
Create table DARY_ARCHIV
(
	id_dar Integer Primary key NOT NULL,
	id_zpusob_platby Smallint,
	id_akce Integer,
	id_darce Integer,
	id_typ_daru Smallint NOT NULL,
	datum Datetime NOT NULL,
	castka Money NOT NULL,
	popis Varchar(30),
	cislo_dokladu Integer NOT NULL,
	variabilni_symbol Char(12)
)
Go

create procedure archiv_dary
@id_dar Integer
as
begin
declare @existuje Integer
set @existuje=0
select @existuje=id_dar from DAR where id_dar=@id_dar
if(@existuje=0)
  begin
   print 'Zadali jste špatné ID daru.'
   return
  end
```

```
else
begin
insert into DARY_ARCHIV(id_dar, id_zpusob_platby, id_akce, id_darce, id_typ_daru,
datum, castka, popis, cislo_dokladu, variabilni_symbol)
select d.id_dar, d.id_zpusob_platby, d.id_akce, d.id_darce, d.id_typ_daru,
d.datum, d.castka, d.popis, d.cislo_dokladu, d.variabilni_symbol
from DAR d where d.id_dar = @id_dar
end
end
go

begin transaction TZ1
execute archiv_dary '1'
select * from DARY_ARCHIV
rollback transaction TZ1
go
```

3.4.1.2 Dárce x Dar

Procedura s vnořeným kurzorem. Má za úkol vypsat kolik dárce poskytl darů a co mu za to náleží. Vstupním parametrem je počet darů, od kterého chceme výpis uskutečnit.

```
create procedure darce_dar
@od Integer
as
BEGIN
  declare kurzor cursor for
    select d.jmeno, d.prijmeni, d.nazev_organizace, count(da.id_dar) as
'pocet_daru', case
        when count(da.id_dar) <= 2 then 'Děkovný dopis'
        when count(da.id_dar) = 3 then 'Pozvánka na aukci'
        when count(da.id_dar) = 4 then 'Pozvánka aukce + DDO'
        when count(da.id_dar) > 5 then 'Status VIP' end as 'popis'
        from DARCE d left join DAR da on d.id_darce = da.id_darce
        group by d.jmeno, d.prijmeni, d.nazev_organizace having count(da.id_dar)
>= @od

        declare @jmeno Varchar(20),
                @prijmeni Varchar(30),
                @nazev_organizace Varchar(30),
                @pocet_daru Integer,
                @popis Varchar(30)

      open kurzor
      fetch next from kurzor into @jmeno, @prijmeni, @nazev_organizace,
@pocet_daru, @popis
      while @@FETCH_STATUS = 0
            begin
            print 'Dárce ' +@jmeno+ ' '+@prijmeni +' ' +@nazev_organizace + '
poskytl dar: '+ convert(varchar(2),@pocet_daru)+ 'x , za což mu náleží:
'+@popis+'.'
      fetch next from kurzor into @jmeno, @prijmeni, @nazev_organizace,
@pocet_daru, @popis
```

```
            end
        close kurzor
        deallocate kurzor
END
go
```

Příklad spuštění procedury:

```
        execute darce_dar 1
```

Výstup:

```
        Dárce - - Firma, s.r.o. poskytl dar: 6x , za což mu náleží: Status VIP.
        Dárce Petr Novák - poskytl dar: 1x , za což mu náleží: Děkovný dopis.
        Dárce Veronika Svobodová - poskytl dar: 1x , za což mu náleží: Děkovný dopis.
```

3.4.1.3 Nový dar

Procedura s devíti vstupními parametry, která má za úkol uložit nový dar. Pro kontrolu duplicity slouží číslo dokladu, jelikož jedno číslo dokladu musí náležet právě jednomu poskytnutému daru. Pokud se dar se vstupním totožným číslem dokladu v databázi nachází, nebude tedy uložen. Pro kontrolu procedura vytiskne ID uloženého daru. Zkouška procedury je spuštěná přes odvolanou transakci.

```
create procedure uloz_dar
  @id_zpusob_platby Smallint,
  @id_akce Integer,
  @id_darce Integer,
  @id_typ_daru Smallint,
  @id_ucel_daru Smallint,
  @castka Money,
  @popis Varchar(30),
  @cislo_dokladu Integer,
  @variabilni_symbol Char(12),
  @id_dar Integer output
as
begin
  set @id_dar = (select top 1 id_dar
    from DAR
    where @cislo_dokladu = cislo_dokladu)
  if (@id_dar is null)
  begin
    insert into DAR (id_zpusob_platby, id_akce, id_darce, id_typ_daru,
id_ucel_daru, datum, castka, popis, cislo_dokladu, variabilni_symbol)
    values (@id_zpusob_platby, @id_akce, @id_darce, @id_typ_daru, @id_ucel_daru,
getdate(), @castka, @popis, @cislo_dokladu, @variabilni_symbol)
     set @id_dar = @@identity
```

```
  end
  else
   begin
    print 'Zadané číslo dokladu má již použit dar s ID:'
   end
end
go


begin transaction TZ3
declare @id_dar Integer
execute uloz_dar 1,null,1,1,null,'2000',null,878,null, @id_dar output
print @id_dar
select * from dar
rollback transaction TZ3
go
```

3.4.2 Trigger

Trigger neboli spoušť, definuje činnost, která se spustí nad definovanou tabulkou. Klíčovými slovy spuštění nad tabulkami jsou: INSERT, UPDATE a DELETE.

Následující příklad triggeru je použit na tabulku Kontakty. Pokud z této tabulky vymažeme data pod zadaným ID dárce a ID kontaktu, budou hodnoty automaticky zálohovány do předem vytvořené tabulky Kontakt záloha. Pokud se vstupní hodnoty ID v tabulce nenachází, vytiskne se hláška: „Zadali jste špatné ID dárce nebo špatné ID kontaktu!“. Naopak po správném zadání hodnot se zobrazí hláška: „Smazání proběhlo úspěšně.“.

```
create table KONTAKT_ZALOHA
(
	id_kontakt_zaloha Integer identity (1,1) Primary key,
	id_darce Integer NOT NULL,
	jmeno Varchar(20),
	prijmeni Varchar(30),
	organizace Varchar(30),
	typ_kontaktu Varchar(30),
	poznamka Varchar(100),
	datum_kontaktu Datetime,
	odpoved_datum Datetime,
	datum_smazani Datetime
)
go
```

```
Create trigger smazani_kontaktu on KONTAKTY
for delete
as
begin
       declare @id_darce Integer,
              @id_typ_kontaktu Integer

select  @id_darce = id_darce from deleted
select @id_typ_kontaktu = id_typ_kontaktu from deleted
  if (@id_darce is null
      or @id_typ_kontaktu is null)
         begin
              print 'Zadali jste špatné ID dárce nebo špatné ID kontaktu!'
              return
         end
       begin
         declare @jmeno Varchar(20),
                 @prijmeni Varchar(30),
                 @organizace Varchar(30),
                 @typ_kontaktu Varchar(30),
                 @poznamka Varchar(100),
                 @datum_kontaktu Datetime,
                 @odpoved_datum Datetime
        select @poznamka = poznamka, @datum_kontaktu = datum, @odpoved_datum =
odpoved_datum from deleted
        select @jmeno = jmeno, @prijmeni = prijmeni, @organizace =
nazev_organizace  from DARCE where @id_darce = id_darce
        select @typ_kontaktu = nazev from TYP_KONTAKTU where @id_typ_kontaktu =
id_typ_kontaktu

        insert into KONTAKT_ZALOHA (id_darce, jmeno, prijmeni, organizace,
typ_kontaktu, poznamka, datum_kontaktu, odpoved_datum, datum_smazani)
        values (@id_darce, @jmeno, @prijmeni, @organizace, @typ_kontaktu,
@poznamka, @datum_kontaktu, @odpoved_datum, getDate())
           begin
              print 'Smazání proběhlo úspěšně.'
           end
       end
end
go
```

Příklad smazání kontaktu:

```
       delete from KONTAKTY where id_darce = 1 and id_typ_kontaktu = 1
```

3.4.3 Pohledy

Pomocí pohledů lze vyselektovat vybrané sloupce z několika různých tabulek a tím tedy docílíme zobrazení pouze námi vybraných a potřebných dat.

Pohledy jsou vytvořeny na základě analýzy programu Rolnička, který obsahuje podobné zobrazení dat.

3.4.3.1 Dar info

Pohled nám vypíše všechny důležité informace o veškerých poskytnutých darech.

```
create view dar_info as
select d.id_dar, d.datum, d.castka, d.popis, td.nazev as 'typ_daru', a.nazev as
'akce', zp.nazev as 'platba', ud.nazev as 'ucel'
from DAR d left join TYP_DARU td on d.id_typ_daru = td.id_typ_daru
left join AKCE a on d.id_akce = a.id_akce
left join ZPUSOB_PLATBY zp on d.id_zpusob_platby = zp.id_zpusob_platby
left join UCEL_DARU ud on d.id_ucel_daru = ud.id_ucel_daru
go
```

Výstup:

	id_dar	datum	castka	popis	typ_daru	akce	platba	ucel
1	1	2015-05-14 08:32:52.760	40000,00	NULL	peněžitý	NULL	převodem na účet	Betlém
2	2	2015-05-14 08:32:52.760	10000,00	koupě výrobku	peněžitý	aukce	hotovost	NULL
3	3	2015-05-14 08:32:52.760	25790,00	sbírka v OC Vaňkovka Brno	peněžitý	veřejná sbírka	hotovost	auto
4	4	2015-05-14 08:32:52.760	15487,00	NULL	peněžitý	NULL	převodem na účet	NULL
5	5	2015-05-14 08:32:52.760	10000,00	NULL	peněžitý	NULL	převodem na účet	Betlém
6	6	2015-05-14 08:32:52.760	30000,00	NULL	peněžitý	NULL	převodem na účet	Betlém
7	7	2015-05-14 08:32:52.760	9000,00	NULL	peněžitý	NULL	převodem na účet	Betlém
8	8	2015-05-14 08:32:52.760	5000,00	NULL	peněžitý	NULL	převodem na účet	Betlém
9	9	2015-05-14 08:32:52.760	15000,00	NULL	peněžitý	NULL	převodem na účet	Betlém

Obr. 15: Pohled – Dar info (Zdroj: vlastní zpracování)

3.4.3.2 Dárce info

Tento pohled je podobný jako předchozí, s tím rozdílem, že nám vypíše důležité informace o všech dárcích.

```
create view darce_info as
select d.jmeno + ' ' + d.prijmeni as 'jméno', d.nazev_organizace + ds.nazev as
'organizace', td.nazev as 'typ dárce', p.psc + ' '+ p.mesto + ' ' + d.ulice as
'adresa', d.telefonni_cislo, d.email
from DARCE d left join DARCE_STREDISKO ds on d.id_stredisko = ds.id_stredisko
left join TYP_DARCE td on d.id_typ_darce = td.id_typ_darce
left join PSC p on d.psc = p.psc
go
```

Výstup:

	jmeno	organizace	typ_darce	bydliste	telefonni_cislo	email
1	Petr Novák	NULL	Fyzická os.	69172 Klobouky u Brna Příční 28	777666555	novak@seznam.cz
2	--	NULL	Právnická os.	62700 Brno - Slatina Krajní 45	888998666	firma@email.cz
3	--	Podnik, a.s.Praha	Právnická os.	14300 Praha - Modřany Břízová 87	666555444	podnik@podnik.cz
4	Veronika Svobodová	NULL	Os. s živnostenským listem	71600 Ostrava Prášilova 147	999444555	svobodova@email.cz
5	Klára Ospalá	NULL	Fyzická os.	62700 Brno - Slatina Hlavní 1	124457898	ospala@centrum.cz

Obr. 16: Pohled – Dárce info (Zdroj: vlastní zpracování)

3.4.3.3 Počet darů

Další pohled v sobě obsahuje agregační funkce. Vypíše nám, kolik každý dárce poskytl darů a jaká byla celková částka za všechny poskytnuté dary.

```
create view pocet_daru as
select d.jmeno + ' ' + d.prijmeni as 'jmeno', d.nazev_organizace as 'organizace',
count(da.id_dar) as 'pocet_daru', sum(da.castka) as 'celkova_castka'
from DARCE d left join DAR da on d.id_darce = da.id_darce
group by d.jmeno, d.prijmeni, d.nazev_organizace
go
```

Výstup:

	jmeno	organizace	pocet_daru	celkova_castka
1	--	Firma, s.r.o.	6	109000,00
2	--	Podnik, a.s.	0	NULL
3	Klára Ospalá	-	0	NULL
4	Petr Novák	-	1	10000,00
5	Veronika Svobodová	-	1	15487,00

Obr. 17: Pohled – Počet darů (Zdroj: vlastní zpracování)

3.5 Přínosy návrhu řešení

Cílem této práce bylo vytvoření návrhu databáze pro evidenci dárců a jejich poskytnutých darů. Na základě analýzy byl vytvořen zcela nový návrh evidence „na míru“ organizace. Celá databáze splňuje požadavky normalizace a požadavky investora.

Pro větší efektivitu celé databáze byly nastíněné i procedury, trigger a pohledy, které je možno při implementaci využít.

Jelikož nebyla databáze v organizaci ještě implementována, nemůžeme hned zaznamenat přínos nové databáze. Některé přínosy lze ovšem očekávat. Konkrétně se jedná o:

- Zrychlení a zjednodušení práce v oddělení fundraisingu.
- Snadné zobrazení potřebných údajů v databázi.
- Získání souhrnných dat z databáze.
- Jednoduchá změna údajů u jednotlivých položek.

3.6 Ekonomické zhodnocení

Za běžných podmínek by návrh a realizaci projektu prováděla zkušená firma. Následující tabulka zachycuje přibližnou celkovou cenu realizace celého projektu. Položky čas a cena jsou pouze orientační.

Tab. 3: Náklady na realizaci (Zdroj: vlastní zpracování)

Činnost	**Čas**	**Cena**
Návrh databáze	25 hod.	8 000 Kč
Realizace návrhu	50 hod.	20 000 Kč
Instalace softwaru	8 hod.	2 500 Kč
Celkem:	**83 hod.**	**30 500 Kč**

ZÁVĚR

Cílem bakalářské práce bylo navrhnout databázi pro neziskovou organizaci Diakonie ČCE – středisko Betlém, sídlící v Kloboukách u Brna, která bude poskytovat potřebné informace o dárcích a věnovaných darech.

První část byla věnována teoretickým poznatkům, díky kterým bylo návrh možné provést. Druhá část je zaměřena na krátký popis organizace a na analýzu současného stavu evidence. Oddělení fundraisingu mi dalo všechny potřebné informace a podklady pro vytvoření celkového návrhu. Třetí část práce se již zabývá konečným návrhem, který byl vytvořen na základě analýzy a splňuje tak požadavky organizace. Pro větší efektivitu databáze byla navrhnuta i část fyzické implementace.

Tato bakalářská práce splnila stanovený cíl a přinesla tak organizaci kompletní dokumentaci návrhu, na kterém bude postaven následný vývoj databáze. Dokončený produkt bude umožňovat lepší evidenci potřebných údajů a možnosti získání souhrnných dat.

SEZNAM POUŽITÝCH ZDROJŮ

(1) CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK. *Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází*. Vyd. 1. Brno: Computer Press, 2009, 584 s. ISBN 978-80-251-2328-7.

(2) KOCH, Miloš a Bernard NEUWIRTH. *Datové a funkční modelování*. Vyd. 4., rozš. Brno: Akademické nakladatelství CERM, 2010, 142 s. ISBN 978-80-214-4125-5.

(3) LUHAN, Jan. *Databázové systémy* (přednášky). Brno: Vysoké učení technické, Fakulta podnikatelská, 2014.

(4) GROFF, James R a Paul N WEINBERG. *SQL: kompletní průvodce*. Vyd. 1. Brno: CP Books, 2005, 936 s. ISBN 80-251-0369-2.

(5) ORACLE CORPORATION. *Drawing the Entity-Relationship Diagram*. [online]. [cit. 2015-01-18]. Dostupné z: https://docs.oracle.com/cd/A97335_02/apps.102/a81358/05_dev1.htm.

(6) KŘÍŽ, Jiří a Petr DOSTÁL. *Databázové systémy*. Vyd. 1. Brno: Akademické nakladatelství CERM, 2005, 111 s. ISBN 80-214-3064-8.

(7) SKŘIVAN, Jaromír. *Databáze a jazyk SQL*. [online]. [cit. 2015-01-18]. Dostupné z: http://interval.cz/clanky/databaze-a-jazyk-sql/.

(8) STANEK, William R. *Microsoft SQL Server 2012: kapesní rádce administrátora*. 1. vyd. Brno: Computer Press, 2013, 544 s. ISBN 978-80-251-3797-0.

(9) ŠIMŮNEK, Milan. *SQL: kompletní kapesní průvodce*. Vyd. 1. Praha: Grada, 1999, 247 s. ISBN 80-7169-692-7.

(10) BETLÉM. *Diakonie ČCE: středisko Betlém*. Klobouky u Brna: Betlém, 2007.

(11) ŠEDIVÝ, Marek a Olga MEDLÍKOVÁ. *Úspěšná nezisková organizace*. 2., aktualiz. a dopl. vyd. Praha: Grada, 2011, 155 s. ISBN 978-80-247-4041-6.

(12) JIRRA SOFTWARE S.R.O. *Rolnička* [online]. [cit. 2015-04-13]. Dostupné z: http://www.jirra.cz/.

(13) CRM PRO NEZISKOVKY. *Salesforce CRM* [online]. [cit. 2015-04-13]. Dostupné z: http://www.crmproneziskovky.cz/.

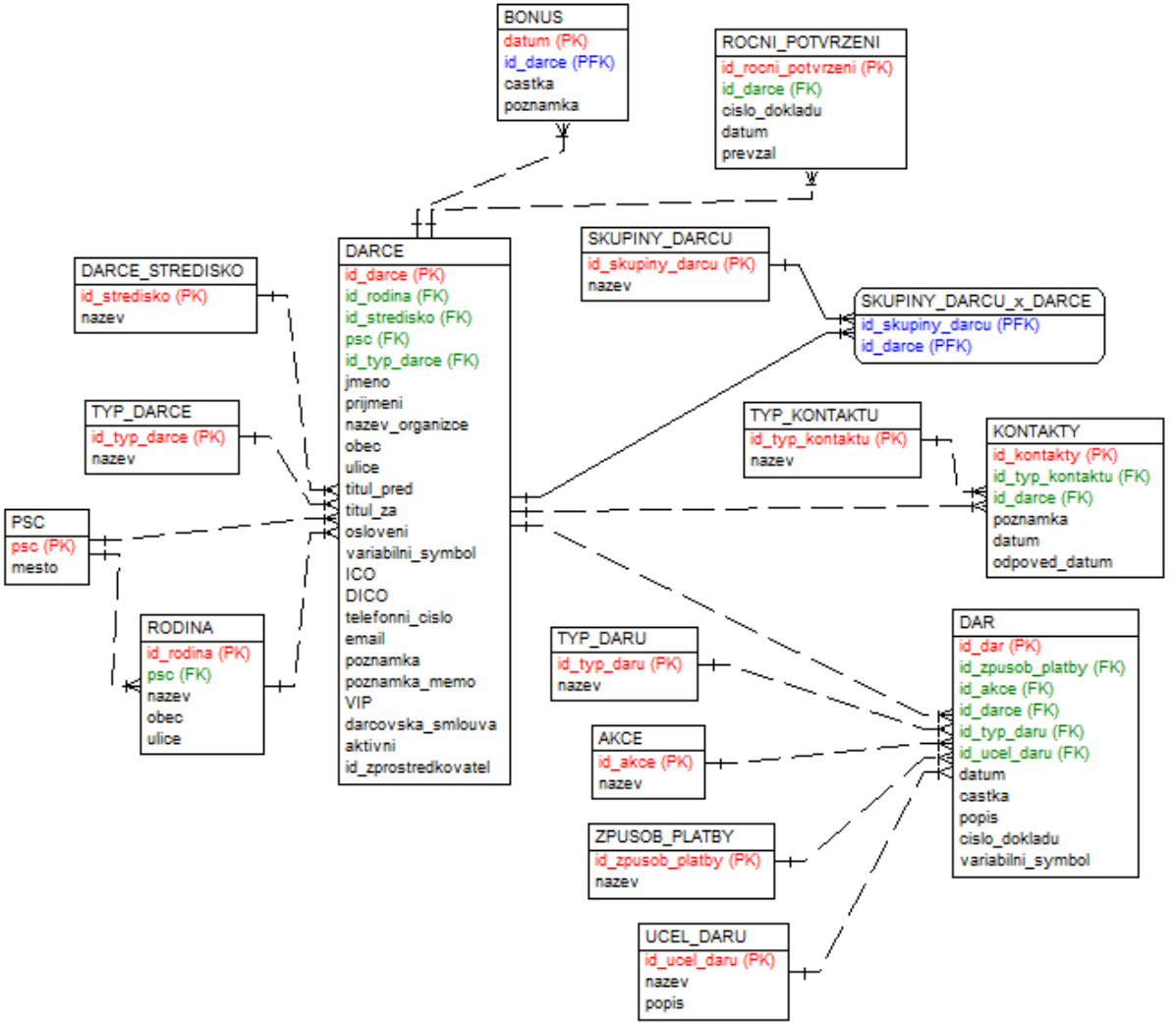
SEZNAM OBRÁZKŮ

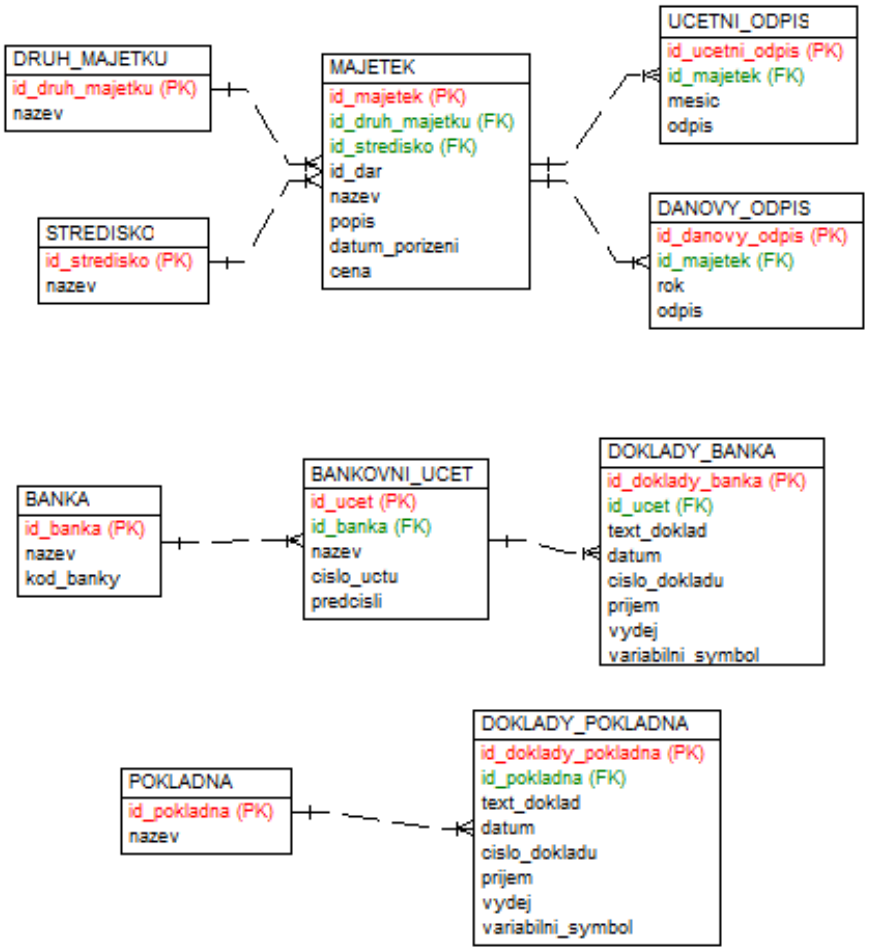
SEZNAM TABULEK

SEZNAM GRAFŮ

SEZNAM PŘÍLOH

Příloha č. 1: E-R diagram




DRUH_MAJETKU
id_druh_majetku (PK)
nazev
STREDISKO
id_stredisko (PK)
nazev
MAJETEK
id_majetek (PK)
id_druh_majetku (FK)
id_stredisko (FK)
id_dar
nazev
popis
datum_porizeni
cena
UCETNI_ODPIS
id_ucetni_odpis (PK)
id_majetek (FK)
mesic
odpis
DANOVY_ODPIS
id_danovy_odpis (PK)
id_majetek (FK)
rok
odpis
BANKA
id_banka (PK)
nazev
kod_banky
BANKOVNI_UCET
id_ucet (PK)
id_banka (FK)
nazev
cislo_uctu
predcisli
DOKLADY_BANKA
id_doklady_banka (PK)
id_ucet (FK)
text_doklad
datum
cislo_dokladu
prijem
vydej
variabilni_symbol
POKLADNA
id_pokladna (PK)
nazev
DOKLADY_POKLADNA
id_doklady_pokladna (PK)
id_pokladna (FK)
text_doklad
datum
cislo_dokladu
prijem
vydej
variabilni_symbol

Příloha č. 2: Datový slovník

TABULKA	ATRIBUTY	DATOVÝ TYP	DÉLKA	PK	INDEX	FK - TABULKA (ATRIBUT)	NOT NULL	POZNÁMKA
DARCE_STREDISKO	**id_stredisko**	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	20				✓	
TYP_DARCE	**id_typ_darce**	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	30				✓	
PSC	**psc**	char	5	✓	✓		✓	
	mesto	varchar	30				✓	
RODINA	**id_rodina**	integer	-	✓	✓			identity (1,1)
	psc	char	5			PSC (psc)		
	nazev	varchar	30				✓	
	obec	varchar	30					
	ulice	varchar	30					

DARCE	**id_darce**	integer	-	✓	✓			identity (1,1)
	id_rodina	integer	-			RODINA (id_rodina)		
	id_stredisko	smallint	-			DARCE_STREDISKO (id_stredisko)		
	psc	char	5			PSC (psc)	✓	
	id_typ_darce	smallint	-			TYP_DARCE (id_typ_darce)	✓	
	jmeno	varchar	20		✓			
	prijmeni	varchar	30		✓			
	nazev_organizace	varchar	30		✓			
	obec	varchar	30					
	ulice	varchar	30				✓	
	titul_pred	varchar	15					
	titul_za	varchar	10					
	osloveni	varchar	20					
	variabilni_symbol	char	12				✓	
	ICO	char	12					
	DICO	char	20					
	telefonni_cislo	integer	-				✓	
	email	varchar	50				✓	
	poznamka	varchar	100					
	poznamka_memo	text	-					
	VIP	bit	-				✓	
	darcovska_smlouva	bit					✓	
	aktivni	bit	-				✓	
	id_zprostredkovatel	varchar	20					
TYP_DARU	**id_typ_daru**	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	20				✓	

AKCE	**id_akce**	integer	-	✓	✓			identity (1,1)
	nazev	varchar	30				✓	
ZPUSOB_PLATBY	**id_zpusob_platby**	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	20				✓	
UCEL_DARU	**id_ucel_daru**	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	30				✓	
	popis	text	-				✓	
DAR	**id_dar**	integer	-	✓	✓			identity (1,1)
	id_zpusob_platby	smallint	-			ZPUSOB_PLATBY (id_zpusob_platby)		
	id_akce	integer	-			AKCE (id_akce)		
	id_darce	integer	-		✓	DARCE (id_darce)		
	id_typ_daru	smallint	-			TYP_DARU (id_typ_daru)	✓	
	id_ucel_daru	smallint	-			UCEL_DARU (id_ucel_daru)		
	datum	datetime	-				✓	
	castka	money	-				✓	
	popis	varchar	30					
	cislo_dokladu	integer	-				✓	
	variabilni_symbol	char	12					
BONUS	datum	datetime	-	✓	✓		✓	
	id_darce	integer	-	✓	✓	DARCE (id_darce)	✓	
	castka	money	-				✓	
	poznamka	varchar	100					
SKUPINY_DARCU	**id_skupiny_darcu**	integer	-	✓	✓			identity (1,1)
	nazev	varchar	20				✓	
SKUPINY_DARCU_x_DARCE	**id_skupiny_darcu**	integer	-	✓	✓		✓	
	id_darce	integer	-	✓	✓		✓	

TYP_KONTAKTU	**id_typ_kontaktu**	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	30				✓	
KONTAKTY	**id_kontakty**	integer	-	✓	✓			identity (1,1)
	id_typ_kontaktu	smallint	-			TYP_KONTAKTU (id_typ_kontaktu)	✓	
	id_darce	integer	-			DARCE (id_darce)	✓	
	poznamka	varchar	100				✓	
	datum	datetime	-				✓	
	odpoved_datum	datetime	-					
ROCNI_POTVRZENI	**id_rocni_potvrzeni**	integer	-	✓	✓			identity (1,1)
	id_darce	integer	-			DARCE (id_darce)	✓	
	cislo_dokladu	integer	-				✓	
	datum	datetime	-				✓	
	prevzal	bit	-				✓	
BANKA	id_banka	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	20				✓	
	kod_banky	varchar	4				✓	
BANKOVNI_UCET	**id_ucet**	smallint	-	✓	✓			identity (1,1)
	id_banka	smallint	-			BANKA (id_banka)	✓	
	nazev	varchar	20				✓	
	cislo_uctu	varchar	15				✓	
	predcisli	varchar	9					
POKLADNA	**id_pokladna**	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	20				✓	

DOKLADY_BANKA	**id_doklady_banka**	smallint	-	✓	✓			identity (1,1)
	id_ucet	smallint	-			BANKOVNI_UCET (id_ucet)	✓	
	text	varchar	20					
	datum	datetime	-				✓	
	cislo_dokladu	integer	-				✓	
	prijem	money	-					
	vydej	money	-					
	variabilni_symbol	varchar	12				✓	
DOKLADY_POKLADNA	**id_doklady_pokladna**	smallint	-	✓	✓			identity (1,1)
	id_pokladna	smallint	-			POKLADNA (id_pokladna)	✓	
	text	varchar	20					
	datum	datetime	-				✓	
	cislo_dokladu	integer	-				✓	
	prijem	money	-					
	vydej	money	-					
	variabilni symbol	varchar	12					
DRUH_MAJETKU	**id_druh_majetku**	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	20				✓	
STREDISKO	**id_stredisko**	smallint	-	✓	✓			identity (1,1)
	nazev	varchar	20				✓	

MAJETEK	**id_majetek**	integer	-	✓	✓		✓	
	id_druh_majetku	smallint	-			DRUH_MAJETKU (id_druh_majetku)	✓	
	id_stredisko	smallint	-			STREDISKO (id_stredisko)	✓	
	id_dar	integer	-		✓		✓	
	nazev	varchar	20				✓	
	popis	test	-					
	datum_porizeni	datetime	-				✓	
	cena	money	-				✓	
UCETNI_ODPIS	**id_ucetni_odpis**	integer	-	✓	✓			identity (1,1)
	id_majetek	integer	-		✓	MAJETEK (id_majetek)	✓	
	mesic	varchar	9				✓	
	odpis	money	-				✓	
DANOVY_ODPIS	**id_danovy_odpis**	integer	-	✓	✓			identity (1,1)
	id_majetek	integer	-		✓	MAJETEK (id_majetek)	✓	
	rok	smallint	-				✓	
	odpis	money	-				✓	

Příloha č. 3: Zdrojový kód

```
create database EVIDENCE_DARCU

use EVIDENCE_DARCU
go

-- VYTVOŘENÍ TABULEK --

Create table DARCE_STREDISKO
(
	id_stredisko Smallint identity (1,1) Primary key,
	nazev Varchar(20) NOT NULL
)
go

Create table TYP_DARCE
(
	id_typ_darce Smallint identity (1,1) Primary key,
	nazev Varchar(30) NOT NULL
)
go

Create table PSC
(
	psc Char(5) Primary key NOT NULL,
	mesto Varchar(30) NOT NULL
)
go

Create table RODINA
(
	id_rodina Integer identity (1,1) Primary key,
	psc Char(5),
	nazev Varchar(30) NOT NULL,
	obec Varchar(30),
	ulice Varchar(30),
Foreign key (psc) references PSC(psc)
)
go

Create table DARCE
(
	id_darce Integer identity (1,1) Primary key,
	id_rodina Integer,
	id_stredisko Smallint,
	psc Char(5) NOT NULL,
	id_typ_darce Smallint NOT NULL,
	jmeno Varchar(20),
	prijmeni Varchar(30),
	nazev_organizace Varchar(30),
	obec Varchar(30),
	ulice Varchar(30) NOT NULL,
	titul_pred Varchar(15),
	titul_za Varchar(10),
	osloveni Varchar(20),
```

```
	variabilni_symbol Char(12) NOT NULL,
	ICO Char(12),
	DICO Char(20),
	telefonni_cislo Integer NOT NULL,
	email Varchar(50) NOT NULL,
	poznamka Varchar(100),
	poznamka_memo Text,
	VIP Bit NOT NULL,
	darcovska_smlouva Bit NOT NULL,
	aktivni Bit NOT NULL,
	id_zprostredkovatel Varchar(20),
Foreign key (id_rodina) references RODINA(id_rodina),
Foreign key (id_stredisko) references DARCE_STREDISKO(id_stredisko),
Foreign key (psc) references PSC(psc),
Foreign key (id_typ_darce) references TYP_DARCE(id_typ_darce),
)
go

Create Index Ijmeno on DARCE(jmeno)
Create Index Iprijmeni on DARCE(prijmeni)
Create Index Iorganizace on DARCE(nazev_organizace)
go

Create table TYP_DARU
(
	id_typ_daru Smallint identity (1,1) Primary key,
	nazev Varchar(20) NOT NULL
)
go

Create table AKCE
(
	id_akce Integer identity (1,1) Primary key,
	nazev Varchar(30) NOT NULL,
)
go

Create table ZPUSOB_PLATBY
(
	id_zpusob_platby Smallint identity (1,1) Primary key,
	nazev Varchar(20) NOT NULL
)
go

Create table UCEL_DARU
(
	id_ucel_daru Smallint identity (1,1) Primary key,
	nazev Varchar(30) NOT NULL,
	popis Text NOT NULL
)
go
```

```
Create table DAR
(
	id_dar Integer identity (1,1) Primary key,
	id_zpusob_platby Smallint,
	id_akce Integer,
	id_darce Integer,
	id_typ_daru Smallint NOT NULL,
	id_ucel_daru Smallint,
	datum Datetime NOT NULL,
	castka Money NOT NULL,
	popis Varchar(30),
	cislo_dokladu Integer NOT NULL,
	variabilni_symbol Char(12),
Foreign key (id_zpusob_platby) references ZPUSOB_PLATBY(id_zpusob_platby),
Foreign key (id_akce) references AKCE(id_akce),
Foreign key (id_darce) references DARCE(id_darce),
Foreign key (id_typ_daru) references TYP_DARU(id_typ_daru),
Foreign key (id_ucel_daru) references UCEL_DARU(id_ucel_daru)
)
go

Create Index Iid_darce on DAR(id_darce)
go

Create table BONUS
(
	datum Datetime NOT NULL,
	id_darce Integer NOT NULL,
	castka Money NOT NULL,
	poznamka Varchar(100),
Foreign key (id_darce) references DARCE(id_darce),
Primary key (datum, id_darce)
)
go

Create table SKUPINY_DARCU
(
	id_skupiny_darcu Integer identity (1,1) Primary key,
	nazev Varchar(20) NOT NULL
)
go

Create table SKUPINY_DARCU_x_DARCE
(
	id_skupiny_darcu Integer NOT NULL,
	id_darce Integer NOT NULL,
Foreign key (id_skupiny_darcu) references SKUPINY_DARCU(id_skupiny_darcu),
Foreign key (id_darce) references DARCE(id_darce),
Primary key (id_skupiny_darcu,id_darce)
)
go


Create table TYP_KONTAKTU
(
	id_typ_kontaktu Smallint identity (1,1) Primary key,
	nazev Varchar(30) NOT NULL
)
go
```

```
Create table KONTAKTY
(
	id_kontakty Integer identity (1,1) Primary key,
	id_typ_kontaktu Smallint NOT NULL,
	id_darce Integer NOT NULL,
	poznamka Varchar(100) NOT NULL,
	datum Datetime NOT NULL,
	odpoved_datum Datetime
Foreign key (id_typ_kontaktu) references TYP_KONTAKTU(id_typ_kontaktu),
Foreign key (id_darce) references DARCE(id_darce)
)
go

Create table ROCNI_POTVRZENI
(
	id_rocni_potvrzeni Integer identity (1,1) Primary key,
	id_darce Integer NOT NULL,
	cislo_dokladu Integer NOT NULL,
	datum Datetime NOT NULL,
	prevzal Bit NOT NULL,
Foreign key (id_darce) references DARCE(id_darce)
)
go

Create table BANKA
(
	id_banka Smallint identity (1,1) Primary key,
	nazev Varchar(20) NOT NULL,
	kod_banky Char(4) NOT NULL
)
go

Create table BANKOVNI_UCET
(
	id_ucet Smallint identity (1,1) Primary key,
	id_banka Smallint NOT NULL,
	nazev Varchar(20) NOT NULL,
	cislo_uctu Varchar(15) NOT NULL,
	predcisli Varchar(9),
Foreign key (id_banka) references BANKA(id_banka)
)
go

Create table POKLADNA
(
	id_pokladna Smallint identity (1,1) Primary key,
	nazev Varchar(20) NOT NULL
)
go

Create table DOKLADY_BANKA
(
	id_doklady_banka Smallint identity (1,1) Primary key,
	id_ucet Smallint NOT NULL,
	text_doklad Varchar(20),
	datum Datetime NOT NULL,
	cislo_dokladu Integer NOT NULL,
	prijem Money,
	vydej Money,
	variabilni_symbol Varchar(12) NOT NULL,
```

```
Foreign key (id_ucet) references BANKOVNI_UCET(id_ucet)
)
go

Create table DOKLADY_POKLADNA
(
	id_doklady_pokladna Smallint identity (1,1) Primary key,
	id_pokladna Smallint NOT NULL,
	text_doklad Varchar(20),
	datum Datetime NOT NULL,
	cislo_dokladu Integer NOT NULL,
	prijem Money,
	vydej Money,
	variabilni_symbol Varchar(12),
Foreign key (id_pokladna) references POKLADNA(id_pokladna)
)
go

Create table DRUH_MAJETKU
(
	id_druh_majetku Smallint identity (1,1) Primary key,
	nazev Varchar(20) NOT NULL
)
go

Create table STREDISKO
(
	id_stredisko Smallint identity (1,1) Primary key,
	nazev Varchar(20) NOT NULL
)
go

Create table MAJETEK
(
	id_majetek Integer Primary key NOT NULL,
	id_druh_majetku Smallint NOT NULL,
	id_stredisko Smallint NOT NULL,
	id_dar integer NOT NULL,
	nazev Varchar(20) NOT NULL,
	popis Text,
	datum_porizeni Datetime NOT NULL,
	cena Money NOT NULL,
Foreign key (id_druh_majetku) references DRUH_MAJETKU(id_druh_majetku),
Foreign key (id_stredisko) references STREDISKO(id_stredisko)
)
go

Create Index Iid_dar on MAJETEK(id_dar)
go

Create table UCETNI_ODPIS
(
	id_ucetni_odpis Integer identity (1,1) Primary key,
	id_majetek Integer NOT NULL,
	mesic Varchar(9) NOT NULL,
	odpis Money NOT NULL,
Foreign key (id_majetek) references MAJETEK(id_majetek),
)
go
```

```
Create Index Iid_majetek on UCETNI_ODPIS(id_majetek)
go

Create table DANOVY_ODPIS
(
	id_danovy_odpis Integer identity (1,1) Primary key,
	id_majetek Integer NOT NULL,
	rok Smallint NOT NULL,
	odpis Money NOT NULL,
Foreign key (id_majetek) references MAJETEK(id_majetek),
)
go

Create Index Iid_majetek on DANOVY_ODPIS(id_majetek)
go

-- VLOŽENÍ TESTOVACÍCH DAT --

insert into DARCE_STREDISKO (nazev)
values ('Brno'),
	('Praha'),
	('Ostrava')
go

insert into TYP_DARCE (nazev)
values ('Fyzická os.'),
	('Právnická os.'),
	('Os. s živnostenským listem')
go

insert into PSC (psc, mesto)
values ('62700', 'Brno - Slatina'),
	('69172', 'Klobouky u Brna'),
	('14300', 'Praha - Modřany'),
	('61700', 'Brno - Komárov'),
	('71600', 'Ostrava')
go

insert into RODINA (psc, nazev, obec, ulice)
values ('69172', 'Novákova', 'Bohumilice', 'Příční 28'),
	('62700', 'Šťastná', null, 'Hlavní 28')
go

insert into DARCE (id_rodina, id_stredisko, psc, id_typ_darce, jmeno, prijmeni,
nazev_organizace, obec, ulice, titul_pred, titul_za, osloveni, variabilni_symbol,
ICO, DICO, telefonni_cislo, email, poznamka, poznamka_memo, VIP,
darcovska_smlouva, aktivni, id_zprostredkovatel)
values (1, null, '69172', 1, 'Petr', 'Novák', '-', null, 'Příční 28', null, null,
'Vážená rodino', '001', null, null, 777666555, 'novak@seznam.cz', null, null, 0,
0, 1, null),
	(null, null, '62700', 2, '-', '-', 'Firma, s.r.o.', null, 'Krajní 45',
null, null, null, '002', '112233445566', 'CZ112233445566', 888998666,
'firma@email.cz', 'roční potvrzení', 'potvrzení posílat i s děkovným dopisem', 1,
1, 1, null),
	(null, 2, '14300', 2, '-', '-', 'Podnik, a.s.', null, 'Březová 87', null,
null, null, '003', '665544332211', 'CZ665544332211', 666555444,
'podnik@podnik.cz', 'rozční potvrzení', null, 0, 1, 1, null),
	(null, null, '71600', 3, 'Veronika', 'Svobodová', '-', null, 'Prášilova
147', 'Ing.',null, 'Vážená paní', '004', null, null, 999444555,
'svobodova@email.cz', null, null, 0, 0, 1, 'Z223'),
```

```
       (2, null, '62700', 1, 'Klára', 'Ospalá', '-', null, 'Hlavní 1', null,
null, null, '005', null, null, 124457898, 'ospala@centrum.cz', 'zaměstnanec',
null, 0, 0, 1, 'Z045')
go

insert into TYP_DARU (nazev)
values ('hmotný'),
       ('nehmotný'),
       ('peněžitý')
go

insert into AKCE (nazev)
values ('veřejná sbírka'),
       ('aukce'),
       ('prodej výrobků'),
       ('AKCE - nové auto')
go

insert into ZPUSOB_PLATBY (nazev)
values ('převodem na účet'),
       ('hotovost')
go

insert into UCEL_DARU (nazev, popis)
values ('nové středisko', 'pro potřeby nově vytvořeného střediska Mirandie'),
       ('auto', 'peníze na koupi nového auta'),
       ('uzdravný pobyt', 'pro speciální pobyt určený pro klienty organizace'),
       ('Betlém', 'určený pro středisko Betlém')
go

insert into BONUS (datum, id_darce, castka, poznamka)
values (getdate(), 5, '2000', 'bonus ke mzdě')
go

insert into DAR (id_zpusob_platby, id_akce, id_darce, id_typ_daru, id_ucel_daru,
datum, castka, popis, cislo_dokladu, variabilni_symbol)
values (1, null, 2, 3, 4, getdate(), '40000', null, 25, '002'),
       (2, 2, 1, 3, null, getdate(), '10000', 'koupě výrobku', 26, '001'),
       (2, 1, null, 3, 2, getdate(), '25790', 'sbírka v OC Vaňkovka Brno', 27,
null),
       (1, null, 4, 3, null, getdate(), '15487',null, 28, '004'),
       (1, null, 2, 3, 4, getdate(), '10000', null, 25, '002'),
       (1, null, 2, 3, 4, getdate(), '30000', null, 25, '002'),
       (1, null, 2, 3, 4, getdate(), '9000', null, 25, '002'),
       (1, null, 2, 3, 4, getdate(), '5000', null, 25, '002'),
       (1, null, 2, 3, 4, getdate(), '15000', null, 25, '002')
go

insert into SKUPINY_DARCU (nazev)
values ('pozvanka aukce'),
       ('pozvanka DOD'),
       ('roční potvrzení'),
       ('přání nového roku')
go

insert into SKUPINY_DARCU_x_DARCE (id_skupiny_darcu, id_darce)
values (1, 1),
       (1, 2),
       (3, 2),
       (1, 3),
```

```
          (3, 3),
          (4, 4),
          (1, 4)
go

insert into TYP_KONTAKTU (nazev)
values ('dopis'),
       ('e-mail'),
       ('telefon'),
       ('osobní setkání')
go

insert into KONTAKTY (id_typ_kontaktu, id_darce, poznamka, datum, odpoved_datum)
values (1, 1, 'pozvánka na DOD', getdate(), null),
       (2, 3, 'poděkování za dar', getdate(), null),
       (4, 1, 'dohodnuté podmínky o novém daru', getdate(), getdate())
go

insert into ROCNI_POTVRZENI (id_darce, cislo_dokladu, datum, prevzal)
values (1, '25', getdate(), 1),
       (3, '26', getdate(), 0)
go

insert into BANKA (nazev, kod_banky)
values ('Česká spořitelna', '0800'),
       ('ČSOB', '0300')
go

insert into BANKOVNI_UCET (id_banka, nazev, cislo_uctu, predcisli)
values (1, 'Účet 1', '123456789', null),
       (1, 'Účet 2', '987654321', '0200'),
       (2, 'Účet 3', '457893125', null)
go

insert into POKLADNA (nazev)
values ('Betlém 1'),
       ('Betlém 2'),
       ('Arkénie 1')
go

insert into DOKLADY_BANKA (id_ucet, text_doklad, datum, cislo_dokladu, prijem,
vydej, variabilni_symbol)
values (1, null, getdate(), '25','40000', null, '002'),
       (3, null, getdate(), '99', '25000', null, '356')
go

insert into  DOKLADY_POKLADNA (id_pokladna, text_doklad, datum, cislo_dokladu,
prijem, vydej, variabilni_symbol)
values (1, null, getdate(), '18', '456', null, null),
       (1, null, getdate(), '1', '100', null, null),
       (1, null, getdate(), '38', null, '50', null)
go

insert into DRUH_MAJETKU (nazev)
values ('auto'),
       ('PC'),
       ('invalidní vozík')
go
```

```
insert into STREDISKO (nazev)
values ('Betlém'),
       ('Arkénie'),
       ('Mirandie')
go

insert into MAJETEK (id_majetek, id_druh_majetku, id_stredisko, id_dar, nazev,
popis, datum_porizeni, cena)
values (256, 1, 1, 25, 'ŠKODA Fabia', 'BVJ 45-68', getdate(), '265000'),
       (327, 2, 3, 28, 'PC - Acer', 'umístění - kancelář', getdate(), '15000')
go

insert into UCETNI_ODPIS (id_majetek, mesic, odpis)
values (327, 'leden', '100')
go

insert into DANOVY_ODPIS (id_majetek, rok, odpis)
values (256, 2014, '5000')
go


-- PROCEDURY --

-- 1. Procedura vytvoří zálohu zadaného daru do předem vytvořené tabulky, jako
vstupní parametr pro uložení je ID daru.

Create table DARY_ARCHIV
(
       id_dar Integer Primary key NOT NULL,
       id_zpusob_platby Smallint,
       id_akce Integer,
       id_darce Integer,
       id_typ_daru Smallint NOT NULL,
       datum Datetime NOT NULL,
       castka Money NOT NULL,
       popis Varchar(30),
       cislo_dokladu Integer NOT NULL,
       variabilni_symbol Char(12)
)
go

create procedure archiv_dary
@id_dar Integer
as
begin
declare @existuje Integer
set @existuje=0
select @existuje=id_dar from DAR where id_dar=@id_dar
if(@existuje=0)
  begin
   print 'Zadali jste špatné ID daru.'
   return
  end
else
begin

insert into DARY_ARCHIV(id_dar, id_zpusob_platby, id_akce, id_darce, id_typ_daru,
datum, castka, popis, cislo_dokladu, variabilni_symbol)
select d.id_dar, d.id_zpusob_platby, d.id_akce, d.id_darce, d.id_typ_daru,
d.datum, d.castka, d.popis, d.cislo_dokladu, d.variabilni_symbol
```

```
from DAR d where d.id_dar = @id_dar
end
end
go

begin transaction TZ1
execute archiv_dary '1'
select * from DARY_ARCHIV
rollback transaction TZ1
go


-- 2. Procedura vypíše kolik dárce poskytl darů a co mu za to náleží.

create procedure darce_dar
@od Integer
as
BEGIN
  declare kurzor cursor for
    select d.jmeno, d.prijmeni, d.nazev_organizace, count(da.id_dar) as
'pocet_daru', case
        when count(da.id_dar) <= 2 then 'Děkovný dopis'
        when count(da.id_dar) = 3 then 'Pozvánka na aukci'
        when count(da.id_dar) = 4 then 'Pozvánka aukce + DDO'
        when count(da.id_dar) > 5 then 'Status VIP' end as 'popis'
        from DARCE d left join DAR da on d.id_darce = da.id_darce
        group by d.jmeno, d.prijmeni, d.nazev_organizace having count(da.id_dar)
>= @od

        declare @jmeno Varchar(20),
                    @prijmeni Varchar(30),
                    @nazev_organizace Varchar(30),
                    @pocet_daru Integer,
                    @popis Varchar(30)

            open kurzor
            fetch next from kurzor into @jmeno, @prijmeni, @nazev_organizace,
@pocet_daru, @popis
            while @@FETCH_STATUS = 0
              begin
                   print 'Dárce ' +@jmeno+ ' '+@prijmeni +' ' +@nazev_organizace
+ ' poskytl dar: '+ convert(varchar(2),@pocet_daru)+ 'x , za což mu náleží:
'+@popis+'.'
                   fetch next from kurzor into @jmeno, @prijmeni,
@nazev_organizace, @pocet_daru, @popis
              end
            close kurzor
            deallocate kurzor
            END
go

execute darce_dar 1
go
```

```
-- 3. Procedura uloží dárci nový dar a jako výstup pak vytiskne ID nově
vytvořeného daru.

create procedure uloz_dar
  @id_zpusob_platby Smallint,
  @id_akce Integer,
  @id_darce Integer,
  @id_typ_daru Smallint,
  @id_ucel_daru Smallint,
  @castka Money,
  @popis Varchar(30),
  @cislo_dokladu Integer,
  @variabilni_symbol Char(12),
  @id_dar Integer output
as
begin
  set @id_dar = (select top 1 id_dar
    from DAR
    where @cislo_dokladu = cislo_dokladu)
  if (@id_dar is null)
  begin
    insert into DAR (id_zpusob_platby, id_akce, id_darce, id_typ_daru,
id_ucel_daru, datum, castka, popis, cislo_dokladu, variabilni_symbol)
    values (@id_zpusob_platby, @id_akce, @id_darce, @id_typ_daru, @id_ucel_daru,
getdate(), @castka, @popis, @cislo_dokladu, @variabilni_symbol)
     set @id_dar = @@identity
  end
  else
   begin
    print 'Zadané číslo dokladu má již použit dar s ID:'
   end
end
go

begin transaction TZ3
declare @id_dar Integer
execute uloz_dar 1,null,1,1,null,'2000',null,878,null, @id_dar output
print @id_dar
select * from dar
rollback transaction TZ3
go


-- TRIGGER --

-- Trigger (spoušť) zálohuje kontakt dárce do předem vytvořené tabulky
KONTAKT_ZALOHA. Jako vstup zde slouží ID kontaktu a ID dárce.

create table KONTAKT_ZALOHA
(
       id_kontakt_zaloha Integer identity (1,1) Primary key,
       id_darce Integer NOT NULL,
       jmeno Varchar(20),
       prijmeni Varchar(30),
       organizace Varchar(30),
       typ_kontaktu Varchar(30),
       poznamka Varchar(100),
       datum_kontaktu Datetime,
       odpoved_datum Datetime,
       datum_smazani Datetime
```

```
)
go

Create trigger smazani_kontaktu on KONTAKTY
for delete
as
begin
	declare @id_darce Integer,
			@id_typ_kontaktu Integer

select  @id_darce = id_darce from deleted
select @id_typ_kontaktu = id_typ_kontaktu from deleted
  if (@id_darce is null
      or @id_typ_kontaktu is null)
          begin
                  print 'Zadali jste špatné ID dárce nebo špatné ID kontaktu!'
                  return
             end
	begin
	  declare @jmeno Varchar(20),
                  @prijmeni Varchar(30),
                  @organizace Varchar(30),
                  @typ_kontaktu Varchar(30),
                  @poznamka Varchar(100),
                  @datum_kontaktu Datetime,
                  @odpoved_datum Datetime
	 select @poznamka = poznamka, @datum_kontaktu = datum, @odpoved_datum =
odpoved_datum from deleted
	 select @jmeno = jmeno, @prijmeni = prijmeni, @organizace =
nazev_organizace  from DARCE where @id_darce = id_darce
	 select @typ_kontaktu = nazev from TYP_KONTAKTU where @id_typ_kontaktu =
id_typ_kontaktu

	 insert into KONTAKT_ZALOHA (id_darce, jmeno, prijmeni, organizace,
typ_kontaktu, poznamka, datum_kontaktu, odpoved_datum, datum_smazani)
	 values (@id_darce, @jmeno, @prijmeni, @organizace, @typ_kontaktu,
@poznamka, @datum_kontaktu, @odpoved_datum, getDate())
          begin
                  print 'Smazani proběhlo úspěšně.'
             end
	end
end
go

delete from KONTAKTY where id_darce = 1 and id_typ_kontaktu = 1
go


-- POHLEDY --

-- 1. Pohled vypíše všechny důležité informace o veškerých poskytnutých darech.

create view dar_info as
select d.id_dar, d.datum, d.castka, d.popis, td.nazev as 'typ daru', a.nazev as
'akce', zp.nazev as 'platba', ud.nazev as 'účel'
from DAR d left join TYP_DARU td on d.id_typ_daru = td.id_typ_daru
left join AKCE a on d.id_akce = a.id_akce
left join ZPUSOB_PLATBY zp on d.id_zpusob_platby = zp.id_zpusob_platby
left join UCEL_DARU ud on d.id_ucel_daru = ud.id_ucel_daru
go
```

```
-- 2. Pohled vypíše naopak všechny důležité informace o dárci.

create view darce_info as
select d.jmeno + ' ' + d.prijmeni as 'jméno', d.nazev_organizace + ds.nazev as
'organizace',
td.nazev as 'typ dárce', p.psc + ' '+ p.mesto + ' ' + d.ulice as 'adresa',
d.telefonni_cislo, d.email
from DARCE d left join DARCE_STREDISKO ds on d.id_stredisko = ds.id_stredisko
left join
TYP_DARCE td on d.id_typ_darce = td.id_typ_darce left join PSC p on d.psc = p.psc
go


-- 3. Pohled vypíše, kolik každý dárce poskytl darů a jaká byla celková částka za
všechny poskytnuté dary.

create view pocet_daru as
select d.jmeno + ' ' + d.prijmeni as 'jméno', d.nazev_organizace as 'organizace',
count(da.id_dar) as 'počet darů',
sum(da.castka) as 'celková částka'
from DARCE d left join DAR da on d.id_darce = da.id_darce
group by d.jmeno, d.prijmeni, d.nazev_organizace
go
```